Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Franci Kaker

Razvoj aplikacijskega sistema za podporo planinski vpisni knjigi

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Damjan Vavpotič

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00074/2013
Datum: 04.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **FRANCI KAKER**

Naslov: **RAZVOJ APLIKACIJSKEGA SISTEMA ZA PODPORO PLANINSKI VPISNI KNJIGI**

DEVELOPMENT OF APPLICATION SYSTEM TO SUPPORT MOUNTAIN LOGBOOK

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

V okviru diplomske naloge preučite in predstavite zahteve za aplikacijski sistem, ki bo podprl delovanje planinske vpisne knjige, nato pa sistem izdelajte do ravni delujočega prototipa, ki bo podprl izbrane funkcije planinske vpisne knjige. Še posebno se posvetite razvoju funkcij za uporabo s strani planinca, ki naj delujejo na mobilni napravi. V okviru diplomske naloge predstavite tudi uporabljene tehnologije in arhitekturo izdelanega sistema.

Mentor:

doc. dr. Damjan Vavpotič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Franci Kaker, z vpisno številko **63090067**, sem avtor diplomskega dela z naslovom:

Razvoj aplikacijskega sistema za podporo planinski vpisni knjigi

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Damjana Vavpotiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 10. septembra 2013 Podpis avtorja:

Zahvaljujem se družini za podporo v času študija ter mentorju doc. dr. Damjanu Vavpotiču za zaupanje in pomoč pri izdelavi diplomskega dela.

Kazalo

Slike

Tabele

Izvorna koda

Povzetek

Na vrhovih gora se nahaja vpisna knjiga. To je dnevnik vrha gore, kamor se vpisujejo planinci, ki osvojijo njen vrh. Glede na to, da dandanes večina ljudi že uporablja pametne mobilne telefone z GPS sprejemnikom in dostopom do interneta, smo se odločili implementirati mobilno vpisno knjigo vrhov gora.

V okviru diplomske naloge smo implementirali in predstavili aplikacijo MountainTrips, ki vključuje osnovne značilnosti klasične vpisne knjige in osebnega dnevnika vzponov planinca ter predstavlja osnovo za razvoj večjega sistema, ki bi uporabnikom omogočal preprostejše medsebojno povezovanje in deljenje informacij o gorah in gorskih poteh.

Ključne besede: Andriod, OpenShift, Java EE, GPS, vpisna knjiga vrhov gora

Abstract

Mountain logbooks are located on mountain peaks. Mountain logbook is a mountain peak log, where hikers log their ascent. Because nowadays most of the people are already using smart phones with GPS receiver and internet access, we decided to implement a mobile version of the mountain logbook.

In this thesis we have implemented and presented MountainTrips application, which includes basic features of a classical mountain logbook and hiker ascents diary. Application is a basis for the development of a larger system, that will allow users simplified interconnection and sharing informations about mountains and mountain trails.

Key words: Andriod, OpenShift, Java EE, GPS, mountain logbook

Seznam kratic

ADT (ang. Android Developer Tools) - razvojna orodja za Android

API (ang. Application Programming Interface) - aplikacijski programski vmesnik

EPL (ang. Eclipse Public License) - javna licenca razvojnega okolja Eclipse

GPS (ang. Global Positioning System) - sistem za globalno določanje lege

HTTP (ang. Hypertext Transfer Protocol) - protokol za prenos večpredstavnostnih vsebin na spletu

HTTPS (ang. Hypertext Transfer Protocol Secure) - HTTP protokol, ki uporablja SSL ali TLS protokol za varno komunikacijo med strežnikom in odjemalcem

IDE (ang. Integrated Development Environment) - integrirano razvojno okolje

Java EE (ang. Java Platform, Enterprise Edition) - poslovna različica platforme Java

Jave SE (ang. Java Platform, Standard Edition) - standardna različica platforme Java

JSON (ang. JavaScript Object Notation) - oblika zapisa za izmenjavo tekstovnih podatkov

JVM (ang. Java Virtual Machine) - virtualni stroj Jave

NDK (ang. Native Development Kit) - komplet orodij, ki omogoča razvoj določenih segmentov Android aplikacije v programskih jezikih C in C++

ORM (ang. Object Relational Mapping) - objektno relacijska preslikava

PaaS (ang. Platform As A Service) - platforma kot storitev

POJO (ang. Plain Old Java Object) - goli Java objekt

RDBMS (ang. Relational Database Management System) - sistem za upravljanje relacijskih podatkovnih baz

REST (ang. Representational State Transfer) - prenos reprezentativnega stanja

SDK (ang. Software Development Kit) - komplet programskih orodij za razvijanje programske opreme

SMTP (ang. Simple Mail Transfer Protocol) - protokol za prenos elektronske pošte

SOAP (ang. Simple Object Access Protocol) - protokol za izmenjavo strukturiranih XML sporočil med ponudnikom spletne storitve in odjemalcem

SQL (ang. Structured Query Language) - strukturirani poizvedovalni jezik

SSL (ang. Secure Sockets Layer) - kriptografski protokol, ki zagotavlja varno komunikacijo med strežnikom in odjemalcem.

TLS (ang. Transport Layer Security) - kriptografski protokol, naslednik SSL, ki zagotavlja varno komunikacijo med strežnikom in odjemalcem

UDDI (ang. Universal Description, Discovery and Integration) - register spletnih storitev

SEZNAM KRATIC

WSDL (ang. Web Services Description Language) - jezik za opis spletnih storitev

WWW (ang. World Wide Web) - svetovni splet

Poglavje 1

Uvod

Ljudje že od nekdaj radi hodijo v hribe. Razlogov za planinarjenje je veliko. Nekateri izkoristijo planinarjenje za odmik od mestnega vrveža in hitrega tempa življenja, druge privlači čudovita narava in razgled na vrhu planin. Veliko planin ima na vrhu vpisno knjigo, kamor se vpišejo planinci, ko osvojijo njen vrh. Vpisna knjiga je dnevnik vrha gore. Večinoma so vpisne knjige zaprte v železne škatle, ki jih ščitijo pred vremenski vplivi. V škatli je poleg vpisne knjige običajno še pisalo in žig vrha gore. Pogosti so primeri, da v škatli kaj manjka, ali pa je vpisna knjiga poškodovana. To marsikaterega planinca odvrne od vpisa v vpisno knjigo. Hkrati pa če želi planinec beležiti svoj dnevnik vzponov, mora poleg vpisa v vpisno knjigo vrha gore imeti svoj dnevnik vzponov, kamor vpisuje osvojene vrhove.

Mnogo ljudem hiter tempo življenja vzame motivacijo za beleženje dogodkov, kot so vzponi v hribe. Ker živimo v času interneta in pametnih telefonov, med katerimi jih večina že vsebuje GPS sprejemnik, smo se odločili izdelati mobilno aplikacijo, ki povzema glavne značilnosti klasične vpisne knjige, hkrati pa planincem olajša beleženje dnevnika vzponov v hribe.

V okviru diplomske naloge smo izdelali aplikacijo MountainTrips. Aplikacija deluje na pametnih mobilnih telefonih z operacijskim sistemom Android ter predstavlja mobilno vpisno knjigo in osebni dnevnik vzponov planinca.

Poglavje 2

Opis ideje

Klasična vpisna knjiga vrha gore služi kot dnevnik vrha gore, kamor se vpisujejo planinci, ki osvojijo njen vrh. V vpisno knjigo se poleg imena in priimka planinca in datuma osvojitve vrha gore vpiše še podatek, kam je planinec namenjen. Zato so vpisi v vpisno knjigo poleg beleženja, kdo je osvojil vrh gore, pomembni pri reševalnih akcijah in iskanju izgubljenih planincev. V primeru, da se planinec izgubi ali ponesreči ob sestopu iz vrha gore lahko podatek, kam se je odpravil iz vrha gore, pomaga pri iskanju ali reševanju osebe.

Klasične vpisne knjige vrhov gora so zaradi vremenskih vplivov ali pa malomarnega ravnanja planincev pogosto poškodovane. Poleg tega mora planinec, če želi voditi evidenco lastnih vzponov v hribe, voditi osebni dnevnik vzponov.

Ker v času interneta, socialnih omrežij in pametnih mobilnih telefonov opazimo pojav, da veliko uporabnikov omenjenih tehnologij rado deli informacije, kje so in kam so namenjeni, se nam je porodila ideja o mobilni aplikaciji, ki poleg značilnosti klasične vpisne knjige omogoča planincem beleženje dnevnika vzponov v hribe, pregledovanje podatkov o posameznih planinah in pregledovanje podatkov o ostalih planincih, ki so se vpisali v dnevnik vrha gore. Pregledovanje podatkov o ostalih planincih omogoča uporabnikom mobilne vpisne knjige, da stopijo preko elektronske pošte ali

mobilnega telefona v stik z ostalimi planinci, ki so pogosto na gori in se tako iz prve roke pozanimajo o morebitnih neznanih poteh, zahtevnosti poti itd.

Poglavje 3

Obstoječe rešitve

Mobilne aplikacije, ki zajema lastnosti klasične vpisne knjige, pregledovanje podatkov o ostalih planincih in vodenje osebnega dnevnika vzponov nismo zasledili. Pri pregledovanju tržišča smo našli dve aplikaciji. Elektronsko vpisno knjigo portala hribi.net in mobilno aplikacijo Hill Lists. Prva vključuje določene funkcionalnosti klasične vpisne knjige, druga pa določene funkcionalnosti osebnega dnevnika vzponov. Preko spletne strani www.hribi.net se lahko vpišemo v elektronsko vpisno knjigo [4]. Aplikacija Hill Lists pa poleg možnosti pregledovanja informacij o gorah omogoča tudi shranjevanje podatkov o posameznem vzponu [5].

3.1 Elektronska vpisna knjiga portala hribi.net

Spletna stran www.hribi.net vključuje elektronsko vpisno knjigo. V vpisno knjigo se lahko vpišemo z mobilnim telefonom, ki ima vgrajen GPS sprejemnik in dostop do interneta. Vpis v vpisno knjigo poteka preko spletnega brskalnika, preko katerega obiščemo spletno stran www.hribi.net/eknjiga. Na spletno stran se prijavimo z uporabniškim imenom in geslom, ki ju določimo ob prijavi na spletno stran www.hribi.net. Ob uspešni prijavi se samodejno pošljejo podatki o naši lokaciji, na podlagi katerih sistem ugotovi ime planine, na kateri se nahajamo. Nato lokacijo potrdimo in s tem je vpis v vpisno

knjigo zaključen. Ker so posredovani GPS podatki dokaz, da se res nahajamo na vrhu gore, se lahko elektronska vpisna knjiga uporablja tudi za zbiranje elektronskih žigov. Glavna pomanjkljivost takšne implementacije vpisne knjige je v tem, da za vpis potrebujemo mobilni signal in dostop do mobilnega interneta. V gorah je mobilni signal zaradi slabih vremenskih razmer ali odročnosti lokacije pogosto moten ali pa ga sploh ni. V tem primeru je vpis v elektronsko vpisno knjigo spletne strani www.hribi.net neizvedljiv. Aplikacija razvita v okviru diplomske naloge rešuje ta problem z možnostjo lokalnega shranjevanja lokacije. Lokalno shranjene lokacije lahko shranimo na strežnik naknadno, ko imamo dostop do brezžičnega ali mobilnega interneta [4].

3.2 Hill Lists

Za iOS operacijski sistem je Graham Haley razvil aplikacijo Hill Lists. Glavni namen aplikacije je uporabnikom ponuditi podatke o planinah Anglije in Irske. Poleg tega omogoča aplikacija shranjevanje podatkov o posameznem vzponu, obveščanje o trenutni lokaciji preko elektronskih ali SMS sporočil, pregled gora na zemljevidu, dostop do vremenskih podatkov na izbranem območju [5] itd. Aplikacija Hill Lists poleg možnosti shranjevanja podatkov o posameznem vzponu vključuje funkcionalnosti, ki bi jih lahko vključili v nadaljnjem razvoju tudi v aplikacijo MountainTrips.

Poglavje 4

Uporabljene tehnologije in razvojna platforma

Pri načrtovanju aplikacije MountainTrips smo veliko pozornosti namenili izbiri tehnologij, v katerih smo aplikacijo razvili. Izbirali smo tehnologije, ki so zanesljive, preizkušene in hkrati prosto dostopne.

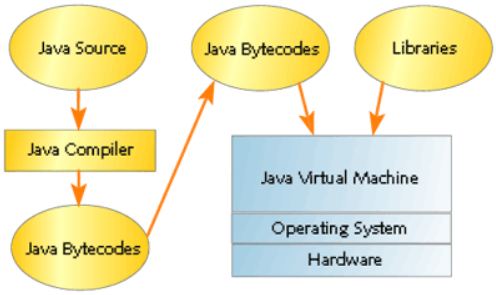
Tudi pri izbiri programskih orodij, s pomočjo katerih smo razvili aplikacijo MountainTrips, je pomemben pogoj predstavljala prosta dostopnost programske opreme.

Ker smo aplikacijo MountainTrips razvili s prosto dostopnimi tehnologijami in programskimi orodji, nismo v razvoj aplikacije MountainTrips vložili nič materialnih sredstev, ampak samo lastno znanje in čas. Zato bi aplikacijo MountainTrips lažje ponudili uporabnikom, kot prosto dostopno programsko opremo.

4.1 Tehnologije strežniškega dela aplikacije MountainTrips

4.1.1 Java

Java je imperativen splošno namenski objektno orientiran programski jezik. Izvorna koda programov napisanih v programskem jeziku Java je shranjena v tekstovnih datotekah s končnico .java. Izvorno kodo programa ne prevedemo neposredno v strojno kodo specifičnega računalnika, ampak jo prevedemo v vmesno bajtno kodo ukazov (ang. bytecode), ki je shranjena v datoteki .class in je razumljiva virtualnemu stroju Jave (kratica JVM) [6]. Virtualni stroj Jave je poseben program, ki interpretira ukaze bajtne kode Jave v strojno kodo ukazov razumljivih specifičnemu računalniku. Ker je virtualni stroj Jave napisan za različne računalniške arhitekture in operacijske sisteme, lahko programe napisane v programskem jeziku Java poganjamo na računalnikih z različno strojno opremo in v različnih operacijskih sistemih [7]. Slika 4.1 prikazuje povezavo med kodo programa, napisanega v programskem jeziku Java, preko virtualnega stroja Jave do različnih operacijskih sistemov.



Slika 4.1: Povezava med kodo programa, napisanega v programskem jeziku Java, preko virtualnega stroja Jave do različnih operacijskih sistemov [8].

Možnost poganjanja programov, napisanih v programskem jeziku Java, na računalnikih z različno strojno opremo in v različnih operacijskih sistemih

omogoča prenosljivost programov brez naknadnega prevajanja ali popravljanja izvorne kode programa. Poleg arhitekturne neodvisnosti in prenosljivosti so pomembne lastnosti programskega jezika Java še preprostost, robustnost in varnost [6].

4.1.2 Java EE

Poslovna različica platforme Jave oz. Java EE je razširitev standardne različice platforme Jave oz. Jave SE. Java EE zagotavlja API in izvajalno okolje za razvoj večjih poslovno informacijskih sistemov in razširja Javo SE s knjižnicami, ki omogočajo objektno relacijske preslikave, spletne storitve itd. Za aplikacije napisane v Javi EE je značilno, da so sestavljene iz posameznih modulov, ki tečejo na aplikacijskem strežniku [9, 10].

Najpomembnejše tehnologije Jave EE so [9]:

- CDI (ang. Context and Dependency Injection)
- EJB (ang. Enterprise Java Beans)
- JMS (ang. Java Message Service)
- JSP (ang. Java Server Pages) in Servleti
- JSF (ang. Java Server Faces)
- JPA (ang. Java Persistence API)
- JCA (ang. Java EE Connector Architecture)
- JNDI (ang. Java Naming and Directory Interface)

4.1.3 Hibernate

Hibernate je knjižnica napisana za programski jezik Java, ki povezuje Java objekte s tabelami v relacijski podatkovni bazi.

Za delo s podatkovno bazo omogoča knjižnica Hibernate namesto neposrednega dostopa do podatkovne baze, s preslikavo objektno usmerjenega podatkovnega modela v klasično relacijsko podatkovno bazo, uporabo visoko nivojskih funkcij. Tako ob menjavi sistema za upravljanje relacijskih podatkovnih baz (kratica RDBMS) nimamo težav [12].

4.1.4 MySQL

Za shranjevanje podatkov, ki jih vnašajo uporabniki aplikacije MountainTrips in podatkov o planinah, smo uporabili odprtokoden sistem za upravljanje z relacijskimi podatkovnimi bazami MySQL. Odprtokodna implementacija podatkovne baze MySQL je popularna izbira podatkovne baze za izdelavo spletnih aplikacij. V določenih segmentih jo uporabljajo tudi spletne strani, kot so Wikipedia, LinkedIn in Twiter [13, 14, 15].

Za uporabo MySQL podatkovne baze smo se odločili, ker jo podpira veliko ponudnikov spletnega gostovanja, med drugim tudi OpenShift PaaS, kjer teče strežniški del aplikacije MountainTrips.

Za dostop do podatkovne baze MySQL obstaja veliko odjemalcev. Mi smo za dostop do podatkovne baze MySQL uporabili program MySQL Workbanch [15]. Operacije nad podatki izvajamo v MySQL podatkovni bazi s poizvedovalnim jezikom SQL.

4.1.5 RESTEasy

Strežniški del aplikacije MountainTrips komunicira z mobilnim delom aplikacije in obratno preko REST spletnih storitev. V strežniškem delu aplikacije smo REST spletne storitve implementirali s pomočjo RESTEasy.

RESTEasy je prenosljiva implementacija specifikacije JAX-RS, JSR-311, ki opisuje Java API za spletne storitve preko protokola HTTP. Spletne storitve, implementirane s pomočjo RESTEasy, lahko poganjamo v vsakem vsebniku (ang. container) servletov [16].

4.1.6 JSON in knjižnica Gson

Podatki med strežnikom in mobilnim delom aplikacije MountainTrips se pošiljajo v obliki JSON. JSON oz. JavaScript zapis objektov (ang. JavaScript Object Notation) je podobno kot XML oblika zapisa za izmenjavo tekstovnih podatkov. JSON oblika zapisa zasede v primerjavi z XML manj prostora,

hkrati pa je hitrejša in preprostejša za parsanje [17]. Izvorna koda 4.1 prikazuje zapis preprostega JSON objekta.

```
{
"planine": [
    {"ime":"Triglav", "višina":"2864"},
    {"ime":"Škrlatica", "višina":"2740"},
    {"ime":"Mangart", "višina":"2679"}
]
}
```

Izvorna koda 4.1: Primer JSON objekta *planine*, ki predstavlja polje treh zapisov *planin*.

JSON uporablja JavaScript sintakso opisovanja podatkovnih objektov. Kljub temu lahko JSON uporabljamo v različnih podatkovnih jezikih [18].

V strežniškem delu aplikacije MountainTrips smo za serializacijo in deserializacijo Java objektov uporabili Googlovo knjižnico Gson [19].

4.1.7 JBoss aplikacijski strežnik

Strežniški del aplikacije MountainTrips lahko poganjamo na JBoss 7 aplikacijskem strežniku, ki implementira poslovno različico platforme Jave (Java EE). Strežnik je napisan v Javi. To pomeni, da ga lahko poganjamo na vseh operacijskih sistemih, ki podpirajo Javo.

JBoss aplikacijski strežnik je odprtokoden in prosto dostopen. Razvili so ga pri podjetju JBoss, ki je zdaj del podjetja Red Hat [20].

Za uporabo JBoss aplikacijskega strežnika sem se odločil, ker ga poznam iz službe in ker ga podpira OpenShift PaaS, kjer poganjamo strežniški del aplikacije MountainTrips.

4.1.8 OpenShift

JBoss aplikacijski strežnik, ki poganja strežniški del aplikacije MountainTrips, teče v oblačni storitvi OpenShift. OpenShift spada v različico platforma kot storitev (kratica PaaS) računalništva v oblaku. Razvijalcem omogoča hiter razvoj in testiranje aplikacij ter nudi gostovanje in po potrebi širitev potrebnih virov za delovanje aplikacije v javnem oblaku [21].

Pri izbiri ponudnika oblačnih storitev smo se odločali med OpenShift PaaS in Google App Engine (kratica GAE). Po našem mnenju imata oba svoje prednosti in slabosti. Spodaj so predstavljena dejstva, ki so vplivala na našo odločitev pri izbiri ponudnika Paas.

Goolgle app engine [25]:

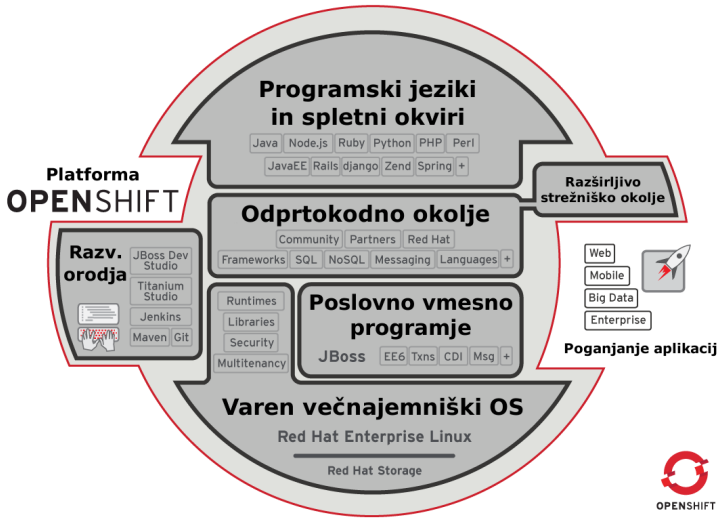
- Zastonj različica GAE ponuja več strežniških virov kot zastonj različica OpenShift PaaS.
- Dobra dokumentacija.
- Zastonj različica GAE uporablja za shranjevanje podatkov ne relacijske podatkovne baze.
- Aplikacije napisane za GAE so relativno težko prenosljive na druge platforme.

OpensShift [22]:

- Omogoča poganjanje aplikacij, ki tečejo na standardnem aplikacijskem strežniku (npr. JBoss AS).
- OpenShift omogoča uporabo tako relacijskih (npr. MySQL) kot tudi ne relacijskih podatkovnih baz (npr. MangoDB).
- Večja prenosljivost aplikacij, saj so napisane za standardne strežnike.

Glede na to, da poznam relacijske podatkovne baze temeljiteje kot ne relacijske in ker poznam tehnologije Jave EE in Jboss AS bolje kot tehnologije GAE, sem se odločil, da za ponudnika PaaS izberem OpenShift.

OpenShift zagotavlja diskovni prostor, procesorske vire, vire delovnega spomina, povezavo z internetom in strežnik Apache ali JBoss AS. Slika 4.2 prikazuje platformo OpenShift.



Slika 4.2: Platforma OpenShift [24].

Oblačna storitev OpenShift PaaS podpira poganjanje spletnih aplikacij v različnih programskih jezikih in spletnih okvirih (ang. web frameworks) ter poganjanje različnih podatkovnih baz. Tabela 4.1 prikazuje tehnologije, ki jih podpira platforma OpenShift.

Programski jeziki	PHP, Python, Ruby, Perl, Node.js, Java
Spletni okviri (ang. web frameworks)	CodeIgniter, CakePHP, Zend, Ruby on Rails, Django, Perl Dancer, Flask, Sinatra, Tornado, web2py
Podatkovne baze	MangoDB, MySQL, PostgreSQL
Strežniki	JBoss EAP, JBoss AS, Tomcat (JBoss EWS)

Tabela 4.1: Programski jeziki, spletni okviri, podatkovne baze in strežniki, ki jih podpira OpenShift PaaS oblačna storitev [22, 23].

4.1.9 Apache Maven

Maven je orodje za gradnjo in upravljanje Java projektov. S pomočjo POM (ang. Project Object Model) datotek opisuje strukturo projekta, njegovo odvisnost z zunanjimi moduli in knjižnicami, nosi podatke o lastniku projekta in definira faze gradnje projekta.

V POM datoteki definiramo odvisnosti (ang. dependenceis) oz. knjižnice, ki jih potrebuje projekt za delovanje. Maven na podlagi definiranih odvisnosti samodejno prenese iz nastavljenih skladišč podatkov (ang. repositories) datoteke, potrebne za gradnjo in delovanje projekta in jih shrani v lokalni predpomnilnik. Definirane odvisnosti lahko Maven prenese iz lokalnih ali javnih skladišč podatkov (npr. Maven 2 Central Repository).

Večje projekte razdelimo na več posameznih modulov. V tem primeru ima vsak modul lastno POM datoteko, ki nosi podatke o gradnji modula in njegovih odvisnostih. S korensko POM datoteko povežemo vse module skupaj. Na ta način omogočimo gradnjo vseh modulov oz. celotnega projekta z enim ukazom [26].

V strežniškem delu aplikacije MountainTrips smo za gradnjo in upravljanje z odvisnostmi aplikacije uporabili Maven, katerega ukaze smo izvajali večinoma preko razvojnega okolja Eclipse, z nameščenim vtičnikom za podporo orodju Maven.

4.2 Spletne storitve

V aplikaciji MountainTrips poteka komunikacija med mobilnim telefonom in strežnikom preko REST spletnih storitev.

Spletne storitve so način komuniciranja med dvema ali več elektronskimi napravami preko svetovnega spleta. Spletna storitev je računalniška funkcija oz. storitev, ki je dostopna preko spletnega naslova in predstavlja uporabno funkcionalnost, ki jo opravlja [28].

Spletne storitve lahko delimo na enostaven dostopni objektni protokol (kratica SOAP) in prenos reprezentacije stanja (kratica REST) [27].

4.2.1 SOAP spletne storitve

SOAP spletne storitve so jasno definiran komunikacijski protokol, za izmenjavo strukturiranih sporočil med ponudnikom spletne storitve in odjemalcem. Podatki med ponudnikom spletne storitve in odjemalcem se pošiljajo v naprej definirani XML obliki. Običajno se podatki pošiljajo preko aplikacijske plasti, najpogosteje preko protokola HTTP, HTTPS ali SMTP.

SOAP sporočilo je sestavljeno iz ovojnice (ang. envelope), ki označi XML dokument, da gre za SOAP sporočilo. Ovojnica je sestavljena iz naslednjih elementov:

- Neobvezna glava (ang. header element) vsebuje podatke, specifične za določeno aplikacijo (podatki za avtentikacijo, podatki o plačilu itd.).
- Obvezno telo (ang. body element) vsebuje informacijo o klicu spletne storitve ali njen odziv.
- Neobvezno sporočilo o napaki (ang. fault element).

SOAP pogosto uporabljamo skupaj z jezikom za opis spletnih storitev (kratica WSDL) in registrom spletnih storitev (kratica UDDI).

SOAP spletne storitve so razširljive, delujejo na različnih protokolih (HTTP, STMTP, TCP, JMS) in so primerne za vse programske modele. Vsi podatki se pri SOAP spletnih storitvah pošiljajo v XML obliki. Kadar pošiljamo velike količine podatkov, je parsanje XML datotek počasno, poleg tega pa predstavljajo pri vsakem XML sporočilu elementi XML sporočila dodatno količino podatkov za prenos [32].

4.2.2 REST spletne storitve

Prenos reprezentacije stanja oz. REST je arhitekturni stil s pomočjo katerega lahko implementiramo REST spletne storitve. Največji sistem, ki ustreza REST arhitekturnemu stilu je svetovni splet (kratica WWW). REST arhitekturni stil je sestavljen iz strežnika in odjemalca. Odjemalec pošlje zahtevek strežniku. Ta ga obdela in vrne odgovor odjemalcu.

Kljub temu, da so sprva načrtovali REST kot del HTTP protokola, REST

ni omejen samo na protokol HTTP, ampak ga lahko implementiramo tudi na drugih protokolih aplikacijske plasti [30].

Glavni cilji REST arhitekture so [30]:

- zanesljivost in razširljivost komuniciranja med komponentami,
- splošnost vmesnikov,
- neodvisna razporeditev in razvoj komponent,
- vmesne komponente med strežnikom in odjemalcem za izboljšanje odzivnosti, varnosti itd.

REST arhitektura vključuje 6 omejitev. REST arhitekturni slog ob upoštevanju omejitev zagotavlja, da ima vsaka implementacija porazdeljenega sistema lastnosti, kot so: hitrost, razširljivost, preprostost, spremenljivost, prenosljivost in zanesljivost [30].

Omejitve REST arhitekturnega sloga [30]:

Odjemalec - strežnik Splošni vmesnik ločuje strežnik do odjemalca. Ločitev omogoča, da se odjemalci ne obremenjujejo s shranjevanjem podatkov na strežniku, strežnik pa ne z uporabniškim vmesnikom in stanjem odjemalca. Na ta način se lahko strežniki in odjemalci razvijajo med sabo ločeno. Odjemalci so prenosljivi, strežniki pa preprostejši in razširljivi.

Brez stanja Strežnik ne shranjuje stanj uporabnikov. To pomeni, da mora zahtevek odjemalca vsebovati podatke o trenutnem stanju in zahtevi, ki jo želi izvesti. Podatki o stanju odjemalca (uporabnika) so shranjeni pri odjemalcu.

Predpomnilnik Odjemalci lahko shranjujejo odgovore v predpomnilnik.

Večslojni sitem Odjemalec je lahko povezan neposredno s strežnikom, kateremu pošilja zahtevke, ali pa preko vmesnih strežnikov. Vmesni strežniki lahko z uravnavanjem prometa in uporabo predpomnilnika izboljšajo zanesljivost sistema. Slabost vmesnih strežnikov je, da lahko njihova uporaba odpre varnostne luknje.

Koda na zahtevo (opcijsko) Strežnik lahko pošlje odjemalcu programsko kodo (Java applets ali JavaScript) in na ta način začasno razširi uporabnost odjemalca.

Enotni vmesnik Uporaba enotnega vmesnika med odjemalcem in strežnikom poenostavi arhitekturo in loči odjemalca od strežnika. To omogoča neodvisen razvoj strežnika in odjemalca.

Glavne značilnosti spletnega vmesnika REST spletnih storitev so [30, 31]:

- storitve oz. funkcije REST spletnih storitev imajo skupni korenski URI naslov spletnega vmesnika (prim. http://mountains.com/MountainTrips-web/rest/),
- spletni vmesnik definira vrsto podatkovnega tipa, ki ga pošilja (ang. internet media type),
- spletni vmesnik podpira množico operacij, ki jih definira HTTP protokol (npr. GET, PUT, POST, DELETE),
- spletni vmesnik izvaja operacije in ukaze, ki jih dobi v URL obliki.

Del aplikacije MountainTrips se izvaja na mobilnih napravah. To pomeni, da dostop do strežniškega dela aplikacije MountainTrips poteka preko mobilnega dostopa do interneta, ki je v večini primerov plačljiv. REST spletne storitve omogočajo pošiljanje podatkov v JSON obliki. Glede na to, da podatki, predstavljeni v JSON obliki, zasedejo manj prostora kot podatki, predstavljeni v XML obliki in ker je parsanje JSON predstavitve podatkov procesorsko in energijsko učinkovitejše od parsanja XML predstavitve podatkov, smo se odločili, da v aplikaciji MountainTrips implementiramo REST spletne storitve.

4.3 Tehnologije mobilnega dela aplikacije MountainTrips

4.3.1 Android

V prvi polovici devetdesetih so se pojavili prvi mobilni telefoni [33]. Privoščili so si jih lahko le redki, marsikdo pa ni imel potrebe, po uporabi mobilnega telefona, saj sta mu stacionarni telefon in faks izpolnila vse želje in potrebe na področju komuniciranja.

Danes si življenja brez mobilnega telefona marsikdo na zna več predstavljati. Moderni mobilni telefoni omogočajo poleg klicanja tudi funkcije, ki so jih včasih omogočali le računalniki. Veliko ljudi uporablja za pregled dnevnih novic in elektronske pošte kar mobilni telefon. Telefone, ki poganjajo mobilni operacijski sistem, omogočajo nalaganje mobilnih aplikacij in opravljanje različnih opravil, za katere so bili včasih namenjeni računalniki, imenujemo pametni mobilni telefoni oz. majhni žepni računalniki.

Poznamo različne mobilne operacijske sisteme, med katerimi je danes najpopularnejši operacijski sistem Android. Operacijski sistem Android temelji na jedru operacijskega sistema Linux. Android je odprtokoden operacijski sistem za mobilne telefone, ki ga razvija podjetje Google. Izdan je pod licenco Apache. Sprva so Android operacijski sistem razvili izključno za mobilne telefone z zaslonom na dotik, ampak ker je Android operacijski sistem odprtokoden in izdan pod licenco, ki dovoljuje svobodno spreminjanje in njegovo uporabo, ga je veliko podjetij prilagodilo svojim potrebam. Zato danes najdemo operacijski sistem Android na različnih elektronski napravah, kot so: računalniške tablice, digitalne kamere, igralne konzole, televizije [34] itd.

Z razvojem aplikacij za operacijski sistem Android se ukvarja veliko ljudi in podjetij. Aplikacije so večinoma razvite v programskem jeziku Java. Uporabniki lahko naložijo aplikacije preko trgovin za prodajo aplikacij, kot sta Google Play in Amazon Appstore, ali prenesejo APK datoteko s spleta in jo naložijo ročno. Za operacijski sistem Android je veliko aplikacij brezplačnih [34].

Do maja leta 2013 je bilo aktiviranih že 900 milijonov naprav z operacijskim sistemom Android in prenesenih 48 milijard aplikacij. Samo v zadnjem letu je bilo aktiviranih okrog 500 milijonov naprav z operacijskim sistemom Android. Torej več kot 1 milijon aktiviranih naprav na dan. Omenjena dejstva in podatek, da so razvijalci v prvi polovici leta 2013 zaslužili več denarja, kot v celotnem letu 2012 kažejo, da je operacijski sistem Android trenutno najpopularnejši operacijski sitem na področju malih elektronskih naprav [34, 36].

4.3.2 Arhitektura operacijskega sistema Android

Slika 4.3: Arhitektura operacijskega sistema Android [34].

Slika 4.3 prikazuje arhitekturo operacijskega sistema Android. Operacijski sistem Android je sestavljen iz štirih plasti. Vsaka plast nudi različne storitve plasti nad njo in pošilja zahteve v izvedbo nižji plasti pod njo.

Najnižja plast operacijskega sistema Android je jedro operacijskega sistema Linux. Operacijski sistemi Android do različice 3.0 (Honeycomb) uporabljajo različico 2.6 novejši pa različico 3.0 jedra Linux. Jedro Linux vključuje gonilnike za strojno opremo, preko katerih operacijski sistem Android izstavlja strojni opremi ukaze in bere podatke iz nje. V jedru operacijskega sistema Android se izvajajo pomembni procesi, ki upravljajo s pomnilnikom, procesi, dostopom do spleta in varnostjo [34, 35].

Plast nad jedrom predstavljata dve vrsti knjižnic. Knjižnice, ki so na sliki 4.3 pobarvane zeleno, so napisane v C ali C++ programskem jeziku in omogočajo napravi upravljanje z različnimi podatkovnimi tipi. S pomočjo teh knjižnic predvajamo glasbene in video posnetke, omogočajo nam shranjevanje podatkov v SQLite podatkovno bazo, prikazovanje HTML spletnih strani itd. Na tej plasti so poleg knjižnic za upravljanje z različnimi podatkovnimi tipi še knjižnice, ki so potrebne za izvajanje programov, napisanih za operacijski sistem Android. Na sliki so pobarvane z rumeno barvo. Sestavljene so iz Java knjižnic, ki so podobne knjižnicam standardne različice Jave (Java SE) in Dalvik virtualnega stroja. Dalvik VM je poseben tip virtualnega stroja Jave, ki je optimiziran, za poganjanje Java aplikacij v okolju z malo spomina in malo procesorske moči [35].

Plast nad knjižnicami predstavlja aplikacijsko ogrodje, na katerem gradijo razvijalci Android aplikacije. Aplikacijsko ogrodje je sestavljeno iz orodij, ki ponujajo razvijalcu možnost upravljanja z viri podatkov, izvajanje klicev, upravljanje s stanji, v katerih se izvaja naša aplikacija [35] itd.

Aplikacijska plast je najvišja plast operacijskega sistema Android. Na tej plasti se izvajajo aplikacije, ki jih uporabljajo uporabniki mobilne naprave [35].

4.3.3 Android podporna knjižnica

Android operacijski sistem se neprestano razvija. Z razvojem se dodajajo nove funkcionalnosti in vmesniki uporabniškega programa (kratica API). Starejše različice operacijskega sistema Android vseh funkcionalnosti in API-jev ne vključujejo (npr. fragmentov in akcijske vrstice). Zato aplikacija, ki uporablja funkcionalnosti novejših različic operacijskega sistema Android, ne deluje na starejših različicah operacijskega sistema Android.

Uporabo nekaterih funkcionalnosti in API-jev novejših različic operacijskega sistema Android, v starejših različicah, nam omogoča Android podporna knjižnica (ang. Android Support Library) [44].

Med načrtovanjem in razvojem aplikacije, je pametno pokriti čim širši

spekter različic operacijskega sistema Android, na katerem bo aplikacija delovala.

Aplikacija MountainTrips deluje na operacijskem sistemu Android različice 2.2 (Froyo) in novejših. To pomeni, da deluje, glede na podatke v tabeli 4.2, na približno 99% vseh naprav, ki poganjajo operacijski sistem Android. Seveda, če imajo dostop do GPS podatkov in interneta.

Različica	Kodno ime	API	Uporaba
1.6	Donut	4	0.1%
2.1	Eclair	7	1.2%
2.2	Froyo	8	2.5%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	33.0%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	22.5%
4.1.x	Jelly Bean	16	34.0%
4.2.x		17	6.5%

Tabela 4.2: Različice operacijskega sistema z razširjenostjo večjo od 0.1% [45].

4.3.4 ActionBarSherlock

Akcijska vrstica (ang. action bar) je vrstica, ki se običajno nahaja na vrhu Android aplikacije. V akcijsko vrstico lahko vključimo naslov aplikacije oz. naslove različnih pogledov aplikacije in različne funkcionalnosti, kot so navigacija med pogledi aplikacije, iskalnik, različne gumbe, menije [42] itd.

Razvijalci operacijskega sistema Android so akcijsko vrstico vključili v operacijski sistem različice 3.0 oz. v API verzije 11. Android podporna knjižnica ne omogoča uporabo akcijske vrstice na starejših različicah operacijskega sistema Android. Zato smo za podporo naprav s starejšim operacijskim siste-

mom Android, ki še ne vključuje akcijske vrstice, za implementacijo akcijske vrstice uporabili knjižnico ActionBarSherlock. Knjižnica ActionBarSherlock je razširitev Android podporne knjižnice, ki zagotavlja podporo za akcijsko vrstico, od različice operacijskega sistema Android 2.x naprej [38, 42].

4.3.5 SQLite

Za shranjevanje strukturiranih in ponavljajočih se podatkov v okviru mobilnega dela aplikacije MountainTrips smo uporabili SQLite sistem za upravljanje z relacijskimi podatkovnimi bazami, ki je del operacijskega sistema Android [42].

SQLite je vgrajena podatkovna baza, ki za razliko od ostalih SQL podatkovnih baz ne deluje v ločenem strežniškem procesu. Za SQLite podatkovno bazo je značilno, da shranjuje in bere podatke neposredno iz datotek na disku [39].

SQLite je majhna v C programskem jeziku napisana knjižnica, ki implementira večino funkcionalnosti poizvedovalnega jezika SQL. Ker je preprosta in ker potrebuje za delovanje malo delovnega spomina, se pogosto uporablja v elektronskih napravah, kot so mobilni telefoni, MP3 predvajalniki [40] itd.

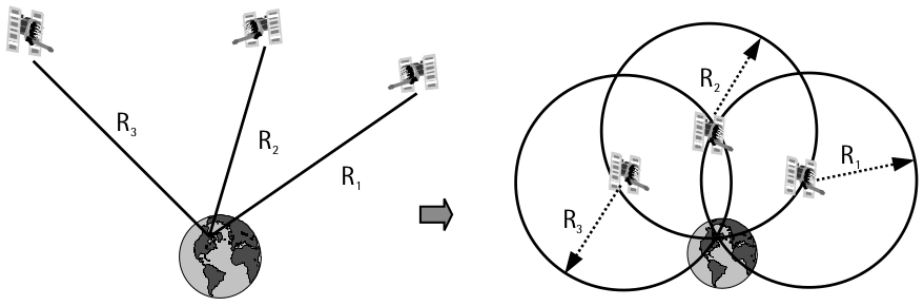
4.4 GPS

GPS je sistem globalnega določanja lege. To je satelitski navigacijski sistem, ki se uporablja za določitev točne lege in časa na Zemlji. Zasnovalo ga je obrambno ministrstvo ZDA, ki ga tudi upravlja. Cena vzdrževanja GPS satelitskega sistema je približno 400 milijonov letno. Sistem GPS je sestavljen iz najmanj 24 satelitov, ki krožijo na višini približno 20200 km nad Zemljo v šestih tirnicah. Vsak satelit obkroži Zemljo dvakrat dnevno in ima nameščeno atomsko uro. Na Zemljo oddaja čas, ki ga razbere iz atomske ure in podatke o tirnici gibanja. Podatke o tirnici gibanja določajo posebne zemeljske opazovalnice [47].

Za pridobitev podatkov o zemljepisni širini, dolžini in nadmorski višini

potrebujemo običajno signal štirih satelitov. Na podlagi razlike med časoma sprejema in oddaje signala ugotovi sprejemnik razdaljo do satelita. Signal med satelitom in sprejemnikom potuje približno s svetlobno hitrostjo [47, 48].

Sprejemnik se nahaja na sferi, s polmerom, ki je enaka razdalji do satelita in katere središče je položaj satelita. Ker sprejemnik sprejema signale več satelitov hkrati, lahko na podlagi presečišč sfer, ki jih določajo posamezni sateliti, izračunamo lego sprejemnika (slika 4.4). Za izračun položaja sprejemnika se najpogosteje uporablja metoda trilateracije [47, 48].



Slika 4.4: Določanje položaja sprejemnika z GPS [46].

Za določitev položaja sprejemnika bi bilo v trirazsežnem prostoru dovolj poznavanje treh sfer. To pomeni, da bi bilo za določitev lokacije sprejemnika dovolj sprejemati signal iz treh satelitov. Ker zahteva postopek izračuna položaja sprejemnika popolno usklajenost ure sprejemnika z urami satelitov, kar je v praksi neizvedljivo, uporabimo za izračun položaja sprejemnika časovni signal dodatnega satelita [48].

Točnost izračuna položaja sprejemnika lahko povečamo z uporabo diferenčne metode, ki temelji na uporabi signalov iz dodatnih virov. Te signale lahko oddajajo oddajniki na znanih lokacijah na Zemlji ali geostacionarni sateliti [48].

GPS satelitski navigacijski sistem lahko uporablja vsak, ki ima v lasti ustrezen GPS sprejemnik. GPS sprejemnike, za sprejem GPS signalov vsebuje danes večina pametnih mobilnih telefonov.

4.5 Razvojno okolje Eclipse

Z razvojnim okoljem Eclipse smo razvili spletni in mobilni del aplikacije MountainTrips. Eclipse je večjezično integrirano razvojno okolje (kratica IDE), s katerim lahko razvijamo aplikacije v različnih programskih jezikih. Osnovna različica razvojnega okolja Eclipse je namenjena razvoju aplikacij v Javi. S pomočjo razširitvenih vtičnikov (ang. an extensible plug-ins) lahko razvojno okolje Eclipse prilagodimo za lažji razvoj aplikacij ali nastavimo za razvoj v različnih programskih jezikih, kot so: Ada, C, C++, PHP, Python, Perl, Ruby, JavaScript, R, Scala, Groovy, Fortran, COBOL, Haskell, Clojure, Scheme in Erlang [1].

Za lažji razvoj spletnega dela aplikacije MountainTrips, smo razširili Eclipse z orodji Jboss (ang. JBoss Tools), za lažji razvoj mobilnega dela aplikacije MountainTrips, pa z Androidovimi razvojnimi orodji (kratica ADT).

Razvojno okolje Eclipse je izdano pod javno licenco Eclipse (kratica EPL). Vsa programska oprema izdana pod javno licenco Eclipse je odprtokodna in prosto dostopna [1].

4.5.1 Orodja JBoss

Orodja JBoss so množica razširitvenih vtičnikov za programsko okolje Eclipse, ki JBoss in Java EE razvijalcem poenostavijo razvoj aplikacij. Olajšajo razvoj aplikacij v tehnologijah, kot so: Hibernate, JBoss AS, JSF, (X)HTML, Seam, Maven [2, 3] itd.

4.5.2 Android SDK

Android aplikacije oz. aplikacije napisane za operacijski sistem Android se večinoma razvijajo v programskem jeziku Java, s pomočjo kompleta programskih orodij, za razvoj Android aplikacij (kratica SDK). Komplet za razvoj Android aplikacij vsebuje razhroščevalnik, knjižnice, dokumentacijo, emulator telefona (slika 4.5), delujoče primere programske kode in navodila za razvoj Android aplikacij. Z uporabo Android NDK (ang. Native Development Kit) lahko določene segmente Android aplikacije razvijamo tudi v programskih jezikih C in C++.

Android SDK deluje na vseh najpopularnejših operacijskih sistemih, kot so Windows, Linux in Mac OS X [37].

Slika 4.5: Emulator operacijskega sistema Android.

4.5.3 Androidova razvojna orodja

Androidova razvojna orodja (kratica ADT) so množica razširitvenih vtičnikov za razvojno okolje Eclipse. Klasičnemu razvojnemu okolju Eclipse dodajo možnost ustvarjanja, pakiranja, nameščanja in razhroščevanja Android aplikacij z uporabo Androidovega programsko razvojnega paketa (kratica SDK) in možnost izvoza APK datotek. Razvojno okolje Eclipse z nameščenim Android ADT vključuje veliko funkcij Android SDK v menijih in oknih razvojnega okolja, vključuje urejevalnika za programski jezik Java in XML ter ima integrirano dokumentacijo za Android API. Razvojno okolje Eclipse z nameščenim Android API omogoča oblikovanje grafičnega vmesnika mobilne aplikacije preko grafičnega vmesnika razvojnega okolja Eclipse.

Razvoj Android aplikacij v razvojnem okolju Eclipse, z nameščenimi razvojnimi orodji za Android, je najhitrejši in najlažji način razvoja Android mobilnih aplikacij [43].

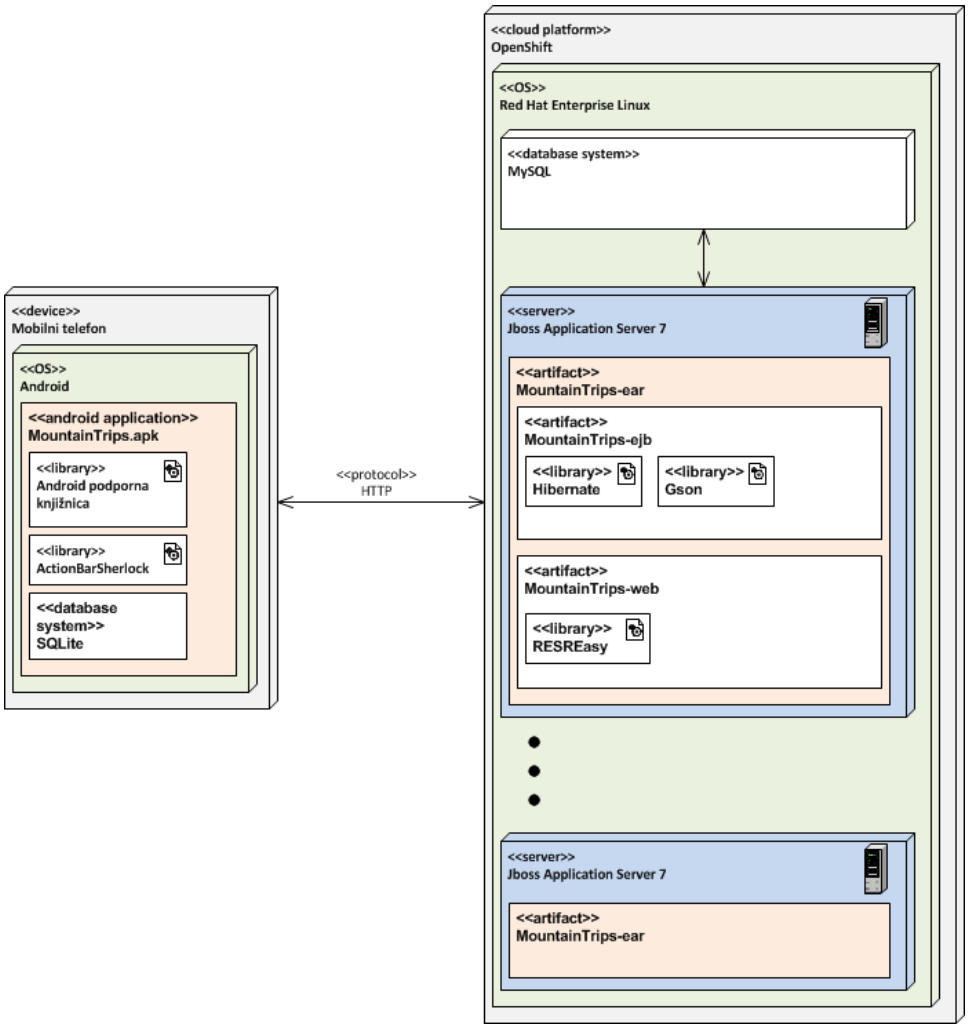
Poglavje 5

Arhitektura aplikacije

Slika 5.1 prikazuje arhitekturo aplikacije MountainTrips. Aplikacijo lahko razdelimo na mobilni del, ki teče na pametnih mobilnih telefonih z operacijskim sistemom Android in strežniški del, ki predstavlja REST spletne storitve in povezavo s podatkovno bazo.

Strežniški del aplikacije se lahko izvaja na samostojnem JBoss 7 aplikacijskem strežniku ali pa v PaaS oblačnih storitvah, ki podpirajo izvajanje aplikacij, napisanih za aplikacijski strežnik JBoos 7. Trenutno strežniški del aplikacije MountainTrips izvaja PaaS oblačna storitev OpenShift.

Izvajanje strežniškega dela aplikacije v PaaS oblačnih storitvah poveča zanesljivost delovanja aplikacije v primeru nenadnega povečanja prometa, saj se v tem primeru aplikacija samodejno razširi na več strežnikov, ki izvajajo zahtevke uporabnikov. Tako ne pride do preobremenitve strežnikov in posledično neodzivnosti v delovanju aplikacije, do česar lahko pride, če se aplikacija izvaja na samostojnem strežniku.



Slika 5.1: Arhitektura aplikacije MountainTrips.

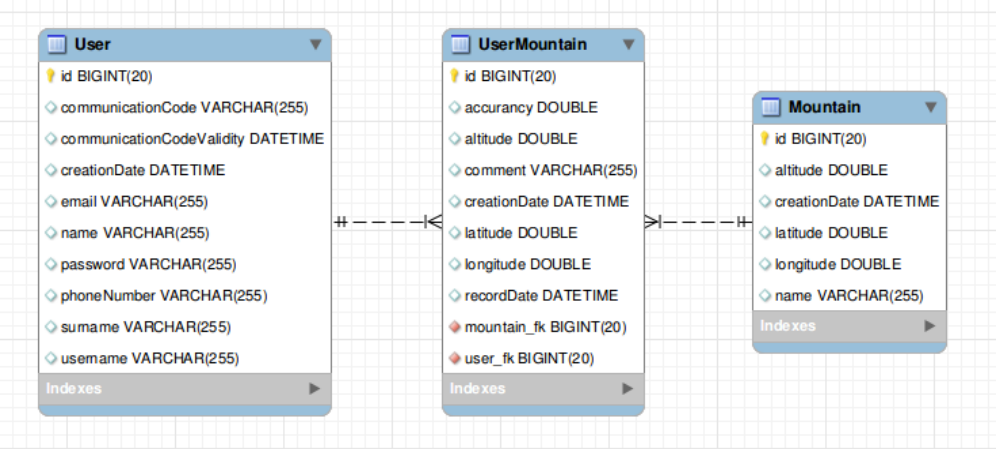
5.1 Podatkovna baza strežniškega dela aplikacije MountainTrips

Aplikacija MountainTrips uporablja knjižnico Hibernate, ki preslika javanske objekte, na podlagi oznak ali XML deskriptorjev, v entitete relacijske podatkovne baze. Aplikacija samodejno ustvari entitete podatkovne baze, na podlagi javanskih objektov, ki hkrati predstavljajo posamezne zapise v podatkovni bazi.

Podatki aplikacije MountainTrips se shranjujejo v MySQL relacijsko podatkovno bazo.

Slika 5.2 predstavlja entitetno relacijski diagram (kratica ER diagram) podatkovne baze aplikacije MountainTrips.

Podatkovna baza aplikacije MountainTrips je sestavljena iz treh med seboj povezanih entitet. V njih shranjujemo podatke o uporabnikih, vpise uporabniških lokacij in podrobnosti o posameznih gorah.



Slika 5.2: ER diagram podatkovne baze aplikacije MountainTrips.

Entiteta User (slo. uporabnik) predstavlja zapise registriranih uporabnikov. Pomen atributov entitete User:

id Id uporabnika.

name Ime uporabnika.
surname Priimek uporabnika.
username Uporabniško ime uporabnika.
email Elektronski naslov uporabnika.
password Geslo uporabnika.
phoneNumber Telefonska številka uporabnika.
creationDate Datum registracije uporabnika.
communicationCode Komunikacijska koda, ki se skupaj z uporabniškim imenom uporablja za avtentikacijo uporabnika.
communicationCodeValidity Datum veljavnosti komunikacijske kode. Ob izračunu komunikacijske kode se nastavi veljavnost 7 dni.

Entiteta Mountain (slo. gora) vsebuje podatke o gorah. Za potrebe razvoja in testiranja aplikacije MountainTrips smo vnesli podatke višjih planin Kamniško Savinjskih Alp, ki smo jih našli na portalu *hribi.net*. Pomen atributov entitete Mountain:

id Id gore.
altitude Nadmorska višina gore.
latitude Zemljepisna širina lokacije gore.
longitude Zemljepisna dolžina lokacije gore.
name Ime gore.
creationDate Datum zapisa podatkov gore v bazo.

Povezava med entitetama User in Mountain je mnogo proti mnogo. Realizirana je s pomočjo vmesne entitete UserMountain, kamor shranjujemo zapise lokacij, ki jih pošiljajo uporabniki iz mobilnega dela aplikacije MountainTrips. Pomen atributov entitete UserMountain:

id Id zapisa v tabeli UserMountain (zapis lokacije uporabnika).
accurancy Natančnost poslanih podatkov o lokaciji.
altitude Nadmorska višina lokacije, kjer se je uporabnik nahajal.
latitude Zemljepisna širina lokacije, kjer se je uporabnik nahajal.
longitude Zemljepisna dolžina lokacije, kjer se je uporabnik nahajal.
comment Komentar.

recordDate Datum, ko se je uporabnik nahajal na lokaciji, ki jo zapis o lokaciji predstavlja.

creationDate Datum zapisa lokacije v bazo.

mountain_fk V primeru, da je shranjena lokacija v območju vrha določene gore, kaže tuji ključ mountain_fk na zapis gore v tabeli Mountain.

user_fk Tuji ključ, ki kaže na zapis uporabnika v tabeli User, kateremu pripada zapis shranjene lokacije.

5.2 Strežniški del aplikacije MountainTrips

Strežniški del aplikacije MountainTrips naložimo na JBoss 7 aplikacijski strežnik v obliki EAR datoteke. EAR datoteka je navadna JAR datoteka s končnico .ear. Vsebuje enega ali več modulov aplikacije, ki se hkrati naložijo na aplikacijski strežnik. Poleg modulov aplikacije vsebuje EAR datoteka še mapo META-INF, s podatki, ki so potrebni za namestitev aplikacije na aplikacijski strežnik [49].

EAR datoteka aplikacije MountainTrips je sestavljena iz dveh modulov:

- MountainTrips-ejb
- MountainTrips-web

5.2.1 Modul MountainTrips-ejb

Modul MountainTrips-ejb predstavlja poslovno logiko aplikacije MountainTrips in izvaja operacije na podatkovni bazi. JBoss aplikacijski strežnik ga naloži in izvaja v EJB 3.1 vsebniku. V EAR datoteki je modul MountainTrips-ejb zapakiran v JAR datoteko.

Razdeljen je na štiri pakete:

- com.mountain.trips.model
- com.mountain.trips.data
- com.mountain.trips.util
- com.mountain.trips.helpers

Paket *com.mountain.trips.model* vsebuje EJB 3.1 objekte (izvorna koda 5.1), ki definirajo strukturo tabel podatkovne baze, njihove instance pa zapise v tabelah podatkovne baze. Imenujemo jih entitete (ang. entity). To so klasični POJO objekti, z dodatnimi oznakami (ang. anotation), ki natančneje opisujejo, kako naj se objekti shranijo oz. preslikajo v podatkovno bazo. Na podlagi entitet knjižnica Hibernate, ki je implementacija JPA 2.0, ustvari podatkovno bazo in s pomočjo katere izvajamo operacije nad podatki v podatkovni bazi.

```
package com.mountain.trips.model;

@Entity
@Table(name = "Mountain")
public class Mountain {
        @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

        private String name;

        public Long getId() {
                return this.id;
        }
        public void setId(Long id) {
                this.id = id;
        }
        public String getName() {
                return this.name;
        }
        public void setName(String name) {
                this.name = name;
        }
}
```

Izvorna koda 5.1: Primer EJB 3.1 entitete.

Podatkovni model aplikacije MountainTrips sestavljajo tri entitete, ki se preslikajo v tabele podatkovne baze:

- Mountain
- User
- UserMountain

Razredi, ki izvajajo operacije nad podatki aplikacije MountainTrips, se nahajajo v paketu *com.mountain.trips.data*.

Poizvedbe po podatkih podatkovne baze se izvajajo preko Criteria programskega vmesnika, ki je del knjižnice Hibernate. Uporaba Criteria programskega vmesnika predstavlja alternativo Hibernate poizvedovalnemu jeziku (kratica HQL) in je priljubljen način izvajanja poizvedb z različnimi parametri [11]. Izvorna koda 5.2 predstavlja preprost primer poizvedbe s pomočjo Criteria programskega vmesnika.

```
Session session = (Session) em.unwrap(Session.class);
Criteria criteria = session.createCriteria(UserMountain.class);
criteria.add(Restrictions.eq("mountain", mountain));
criteria.addOrder(Order.desc("recordDate"));
List<UserMountain> list = criteria.list();
```

Izvorna koda 5.2: Primer Criteria poizvedbe.

Paket *com.mountain.trips.util* vsebuje razred Resources (slo. viri), ki zagotavlja EntityManager vmesnik za komunikacijo s podatkovno bazo in Logger za beleženje različnih dogodkov v aplikaciji.

Zadnji paket modula MountainTrips-jar predstavlja *com.mountain.trips.helpers*. V njem se nahaja razred *HashHelper*. Uporablja se za izračun komunikacijske kode s pomočjo SHA-256 zgoščevalne funkcije.

5.2.2 Modul MountainTrips-web

Spletni del aplikacije MountainTrips predstavlja modul MountainTrips-web. V EAR datoteki je modul MountainTrips-web zapakiran v datoteko tipa

WAR, ki lahko hrani JSP strani, servlete, različne javanske razrede in statične datoteke, kot so klasične HTML strani in CSS datoteke. Poleg omenjenih razredov in datotek vsebuje WAR datoteka še mapo WEB-INF s podatki o strukturi spletnega dela aplikacije. Datoteka *web.xml*, ki se nahaja v mapi WEB_INF, določa servletom poti, preko katerih so dostopni [50].

Spletni del aplikacije MountainTrips vključuje REST spletne storitve izdelane s pomočjo RESTEasy implementacije specifikacije JAX-RS. Sestavljata ga paketa:

- com.mountain.trips.rest
- com.mountain.trips.rest.interceptors

Paket *com.mountain.trips.rest* vsebuje dva razreda. Razreda $UserService$ in $MountainService$ predstavljata različne vire oz. funkcionalnosti spletnega dela aplikacije MountainTrips v obliki REST spletnih storitev. Vsebujeta oznake, ki opredelijo poti do razredov oz. njihovih metod, ter natančneje definirajo dostop in lastnosti posameznega vira spletne storitve.

Vsak razred, ki implementira metode REST spletnih storitev in vsaka metoda, ki predstavlja določeno funkcionalnost spletnega dela aplikacije MountainTrips vsebuje oznako $Path$. Oznaka $Path$ definira relativno pot do vira REST spletne storitve.

Razredi, ki implementirajo metode REST spletnih storitev, se izvajajo v vsebniku servletov (ang. servlet cointainer), zato jim moramo v datoteki web.xml, ki se nahaja v mapi WEB_INF datoteke WAR, nastaviti pot, preko katere so dostopni (izvorna koda 5.3).

```
<servlet-mapping>
    <servlet-name>javax.ws.rs.core.Application</servlet-name>
    <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Izvorna koda 5.3: Nastavitev poti do virov REST spletnih storitev.

Metode, ki predstavljajo funkcionalnosti oz. vire REST spletnih storitev vsebujejo različne oznake. Oznake natančneje opredelijo način dostopa do določenega vira spletne storitve in definirajo tipe parametrov, ki jih metode sprejemajo. Izvorna koda 5.4 predstavlja primer razreda in metode REST spletne storitve.

Oznake, ki natančneje opišejo dostop, do vira REST spletne storitve [29, 51]:

@GET, @PUT, @POST, @DELETE, @HEAD Definirajo tip HTTP zahtevka vira REST spletne storitve.

@Produces Definira vrsto podatkovnega tipa, ki ga vir spletne storitve vrača.

@Consumes Definira vrsto podatkovnega tipa, ki ga vir spletne storitve sprejme.

Oznake, ki definirajo način, po katerem išče določena metoda oz. vir spletne storitve parametre so [29, 51]:

- @PathParam
- @QueryParam
- @MatrixParam
- @HeaderParam
- @CookieParam
- @FormParam
- @DefaultValue
- @Context

```
@Path("/")
public class MountainService {
        @Inject
        private MountainAgent mountainAgent;

        @POST
    @Path("/getMountainName")
    @Produces({ "application/json" })
    public Response getMountainName(
    @FormParam("latitude") String latitude,
        @FormParam("longitude") String longitude,
        @FormParam("altitude") String altitude,
        @FormParam("accurancy") String accuracy) {
                Response.ResponseBuilder builder = null;
                Map<String, String> responseObj = new HashMap<
                    String, String>();

                responseObj.put("mountainName", mountainAgent.
                    getMountainName(latitude, longitude));

                builder = Response.status(Response.Status.
                    ACCEPTED).entity(responseObj);
                return builder.build();
    }

}
```

Izvorna koda 5.4: Primer razreda in metode REST spletne storitve.

V paketu *com.mountain.trips.rest.interceptors* se nahaja razred *SecurityInterceptor*. Razred prestreže klice metod, ki zahtevajo avtentikacijo uporabnika z uporabniškim imenom in geslom oz. komunikacijsko kodo. Če so podatki za avtentikacijo napačni, se klic metode oz. vira spletne storitve ne izvede.

5.2.3 Avtentikacija uporabnika

Določene funkcionalnosti spletnega dela aplikacije MountainTrips so dostopne samo uporabnikom, ki so ustrezno avtenticirani oz. prijavljeni v aplikacijo.

Ena od značilnosti REST spletnih storitev je, da so brez stanja (ang. stateles). Vsak zahtevek odjemalca mora vsebovati vse potrebne podatke za izvedbo zahteve. Torej tudi podatke za avtentikacijo uporabnika. Najpreprostejša rešitev avtentikacije preko REST spletnih storitev je, da zahtevkom, ki za izvedbo potrebujejo avtentikacijo, pripnemo uporabniško ime in geslo. V primeru veljavnega uporabniškega imena in gesla strežnik zahtevek izvede in vrne rezultat, v primeru napačnega uporabniškega imena in gesla pa strežnik zahtevek zavrne oz. ga ne izvede.

Pošiljanje uporabniškega imena in gesla preko HTTP protokola v vsakem zahtevku odjemalca odpira potencialno varnostno luknjo. HTTP zahtevke je relativno preprosto prestreči. V primeru, da napadalec prestreže uporabniško ime in geslo, dobi napadalec celoten dostop do uporabniškega profila.

V aplikaciji MountainTrips smo pomanjkljivosti pošiljanja uporabniškega imena in gesla preko HTTP protokola v vsakem zahtevku odjemalca deloma odpravili z uvedbo komunikacijske kode, ki se ustvari in pošlje uporabniku, ko se le ta prijavi.

Komunikacijska koda ima podobno vlogo kot ID seje (ang. session ID) na spletnih straneh, ki imajo prijavo oz. avtentikacijo uporabnika realizirano s pomočjo seje. Komunikacijska koda v povezavi z uporabniškim imenom enolično določa uporabnika.

Avtentikacija oz. prijava uporabnika z uporabo komunikacijske kode je preprosta. Odjemalec pošlje za prijavo strežniku uporabniško ime in geslo. Če sta uporabniško ime in geslo pravilna, ustvari strežnik komunikacijsko kodo, ji določi omejeno veljavnost sedmih dni in ustvarjene podatke shrani v podatkovno bazo. Nato strežnik pošlje komunikacijsko kodo odjemalcu. Odjemalec si komunikacijsko kodo lokalno shrani. Tako je uporabnik uporabnik prijavljen in lahko kliče metode REST spletnih storitev, ki zahtevajo avtentikacijo.

Odjemalec vsakemu klicu metode REST spletnih storitev aplikacije MountainTrips pripne uporabniško ime in veljavno komunikacijsko kodo. Strežnik preveri uporabniško ime, komunikacijsko kodo in njeno veljavnost. Če so prejeti podatki pravilni, strežnik zahtevek izvede.

V primeru, da napadalec prestreže uporabniško ime in komunikacijsko kodo, dobi napadalec le začasen dostop do uporabniškega profila. Če se uporabnik odjavi in ponovno prijavi, se ustvari nova komunikacijska koda, ki zamenja staro.

Za preprečitev prestrezanja komunikacijske kode bi lahko implementirali komunikacijo med strežnikom in odjemalcem s pomočjo SSL ali TLS protokola.

5.3 Mobilni del aplikacije MountainTrips

Izkušenj v razvoju aplikacij za operacijski sistem Android pred izdelavo aplikacije MountainTrips v okviru diplomske naloge nismo imeli. Zato nam je načrtovanje mobilnega dela aplikacije MountainTrips predstavljalo še večji izziv. V poglavju Arhitektura mobilnega dela aplikacije MountainTrips so opisane glavne arhitekturne značilnosti mobilnega dela aplikacije MountainTrips.

Pri razvoju mobilnega dela aplikacije MountainTrips smo ugotovili, da je razvoj zanesljive mobilne aplikacije težji in počasnejši kot razvoj spletnih aplikacij. Med razvojem mobilnega dela aplikacije MountainTrips smo naleteli na težave z implementacijo odjemalca REST spletnih storitev, na težave, ki se pojavijo z vrtenjem zaslona, nepopolno združljivostjo novejših programskih vmesnikov s starejšimi različicami Android operacijskega sistema itd.

Mobilni del aplikacije MountainTrips lahko naložimo na vsako mobilno oz. elektronsko napravo, ki omogoča prenos mobilnih podatkov oz. povezavo z internetom, sprejem GPS signala in poganja operacijski sistem Android različice 2.2 ali novejši.

Na mobilno napravo naložimo aplikacijo MountainTrips v obliki APK

datoteke. To je ZIP datoteka s končnico .apk, ki vključuje datoteke, potrebne za delovanje Android aplikacije na mobilni napravi z operacijskim sistemom Android [53].

Izvorna koda Android aplikacije, ki predstavlja mobilni del aplikacije MountainTrips, je razdeljena na osem paketov, ki delijo aplikacijo na funkcionalno logične sklope. Razdelitev Android aplikacije na pakete izboljša preglednost izvorne kode aplikacije.

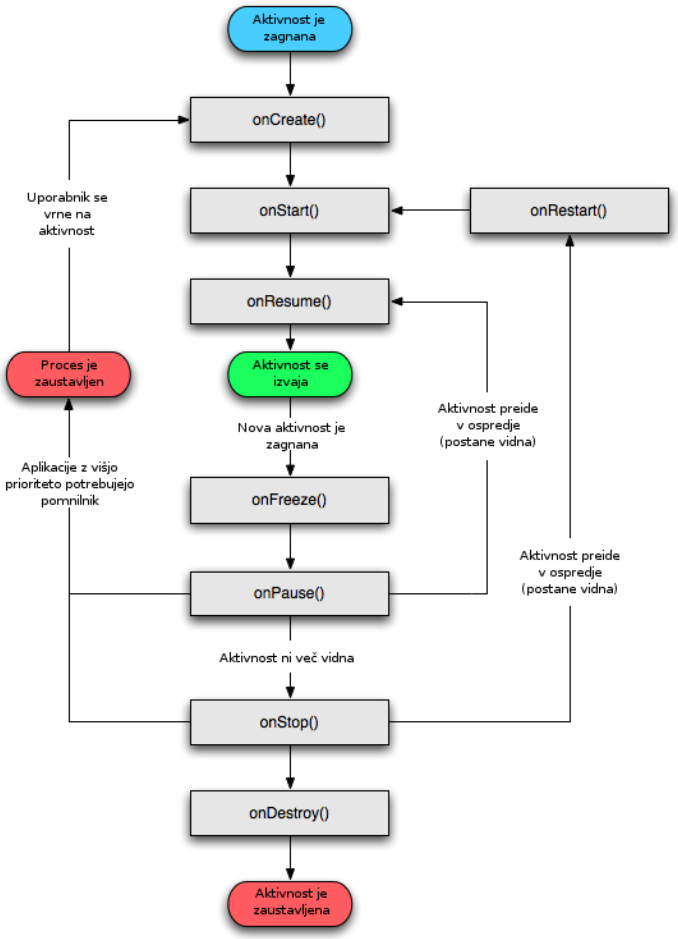
Paketi Android MountainTrips aplikacije:

- com.mountain.trips.mobile.activities
- com.mountain.trips.mobile.activities.arrayadapter
- com.mountain.trips.mobile.activities.elements
- com.mountain.trips.mobile.data
- com.mountain.trips.mobile.helpers
- com.mountain.trips.mobile.receiver
- com.mountain.trips.mobile.service
- com.mountain.trips.mobile.workerfragments

5.3.1 Paket com.mountain.trips.mobile.activities

Paket *com.mountain.trips.mobile.activities* vključuje razrede, ki predstavljajo reprezentativni del mobilnega dela aplikacije MountainTrips. Vsak razred omogoča izvajanje različnih aktivnosti uporabnika, zato imenujemo te razrede aktivnosti aplikacije (ang. activity).

Razredi v paketu *com.mountain.trips.mobile.activities* predstavljajo aktivnosti aplikacije. S pomočjo XML datotek, v mapi */MountainTrips/res/layout* definirajo uporabniški vmesnik in njegovo odzivanje na uporabniške vnose. V njih so implementirane različne metode, ki se prožijo ob vnaprej definiranih dogodkih, kot so: uporabnik odpre pogled določene aktivnosti, uporabnik se pomika med pregledi aplikacije, izvajanje aplikacije prekine telefonski klic, uporabnik zapre aplikacijo [41] itd. Slika 5.3 prikazuje metode, ki se prožijo v različnih stanjih izvajanja aktivnosti.



Slika 5.3: Življenski cikel aktivnosti [52].

5.3.2 Paket com.mountain.trips.mobile.activities.arrayadapter

V paketu *com.mountain.trips.mobile.activities.arrayadapter* so razredi, ki razširjajo programski vmesnik ArrayAdapter. To so razredi, ki priskrbijo podatke seznamu elementov in definirajo izpis posameznih elementov v seznamu podatkov (element ListView). Ker razredi razširjajo funkcionalnosti aktivnosti, je paket *com.mountain.trips.mobile.activities.arrayadapter* vključen v paketu *com.mountain.trips.mobile.activities*.

5.3.3 Paket com.mountain.trips.mobile.activities.elements

Podobno kot v paketu *com.mountain.trips.mobile.activities.arrayadapter* so tudi v paketu *com.mountain.trips.mobile.activities.elements* razredi, ki razširjajo funkcionalnosti aktivnosti. Paket *com.mountain.trips.mobile.activities.elements* vključuje razrede, ki olajšajo uporabo gradnikov uporabniškega vmesnika.

V paketu *com.mountain.trips.mobile.activities.elements* je razred *SimpleElements*, ki je namenjen za enotno prikazovanje obvestil o napakah in dalj časa trajajočih operacijah.

5.3.4 Paket com.mountain.trips.mobile.data

Razredi v paketu *com.mountain.trips.mobile.data* skrbijo za pridobivanje, prenos in shranjevanje podatkov. Poglavje opisuje najpomembnejše razrede paketa *com.mountain.trips.mobile.data*. Poleg opisanih razredov so v paketu *com.mountain.trips.mobile.data* še navadni POJO objekti, ki se uporabljajo za prenos in predstavitev podatkov v aplikaciji MountainTrips.

GpsLocationListener

Razred *GpsLocationListener* implementira metodo *getLocation*() (izvorna koda 5.5), ki vrača GPS podatke o lokaciji uporabnika. Metoda najprej preveri, ali je sprejem GPS signala vključen. Če je, požene sprejemanje GPS signala in čaka podatke o lokaciji. V kolikor traja sprejemanje podatkov več kot tri minute oz. je sprejemanje GPS signala onemogočeno, vrne metoda *getLocation*() vrednost null.

```
public Location getLocation() {
    //Ali je sprejemanje GPS signala vključeno?
    locationManager = (LocationManager) mContext.getSystemService
        (mContext.LOCATION_SERVICE);
    isGPSEnabled = locationManager.isProviderEnabled(
        LocationManager.GPS_PROVIDER);
    if(isGPSEnabled) {
        //Poženemo sprejemanje GPS signala
        locationManager.requestLocationUpdates(LocationManager.
            GPS_PROVIDER, MIN_TIME_BW_UPDATES,
            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
        ...
        //Pridobivanje podatkov o trenutni lokaciji in preverjanje
             njihove ažurnosti.
        location = locationManager.getLastKnownLocation(
            LocationManager.GPS_PROVIDER);
        if(location != null && (System.currentTimeMillis() -
            location.getTime()) < 1000*30) {
            return location;
        }
        ...
    } else {
        // Dostop do GPS-a je onemogočen.
        Log.w(TAG, "Gps ni omogocen.");
    }
    return null;
}
```

Izvorna koda 5.5: Glavni deli metode getLocation().

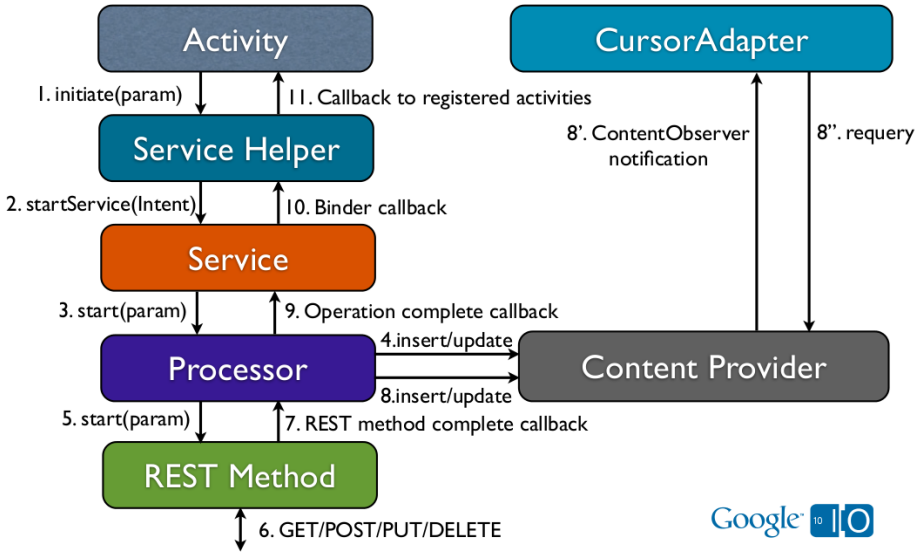
RestClient

Komunikacija med mobilnim in strežniškim delom aplikacije MountainTrips poteka preko REST spletnih storitev. Mobilni del aplikacije predstavlja odjemalca za strežniški del aplikacije MountainTrips.

Posebnih knjižnic za implementacijo komunikacije med mobilnim Android

odjemalcem in strežniškim delom aplikacije preko REST spletnih storitev nismo našli. Na spletu obstajajo knjižnice, ki omogočajo lažjo izvedbo omenjene komunikacije, vendar so večinoma še v razvojni fazi. Ker smo želeli v aplikacijo vključiti le preizkušene tehnologije, ki jim lahko zaupamo, smo REST komunikacijo med strežnikom in mobilnim odjemalcem v mobilnem delu aplikacije MountainTrips implementirali sami.

Leta 2010 je Googlov razvijalec Virgil Dobjanschi na konferenci Google I/O na predavanju o razvoju Android REST odjemalcev (ang. Developing Android REST Client Applications) predstavil tri možne arhitekture razvoja omenjenih Android odjemalcev, njihove implementacije, z morebitnimi spremembami, glede na njihove potrebe, pa prepustil razvijalcem [54, 55].

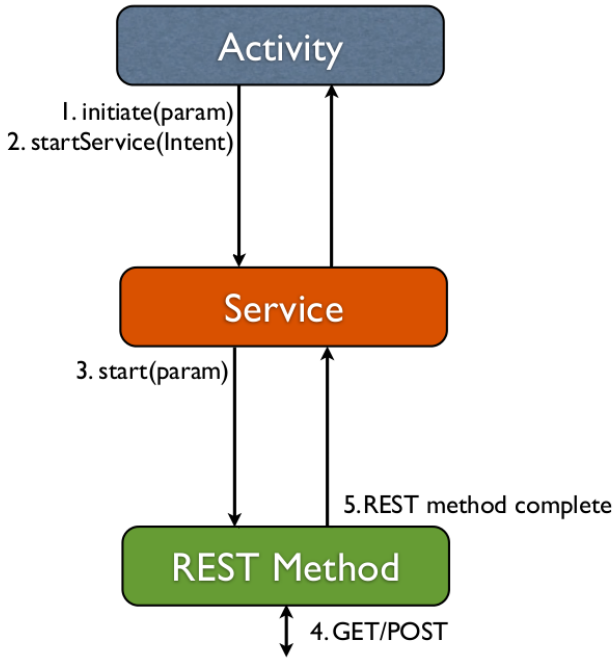


Slika 5.4: Arhitektura Android REST odjemalcev, predstavljena na konferenci Google I/O 2010 [55].

Slika 5.4 prikazuje prvo, od treh predstavljenih arhitektur Android REST odjemalcev. Prikazana arhitektura nam je predstavljala izhodišče za implementacijo Android REST odjemalca.

Celovita implementacija arhitekture Android REST odjemalca, predstavljena na sliki 5.4 zagotavlja pravilno delovanje aplikacije tudi v primeru, da aplikacijo med izvajanjem REST zahtevka prekine bolj pomembna aplikacija oz. opravilo, kot je na primer dohodni klic.

Pri načrtovanju mobilnega dela aplikacije MountainTrips smo sklenili, da bomo težave, ki se pojavijo, če prekine aplikacijo MountainTrips druga, bolj pomembna aplikacija oz. opravilo, reševali ob nadaljnjem razvoju aplikacije. Zato smo implementirali poenostavljeno različico arhitekture, ki je prikazana na sliki 5.5.



Slika 5.5: Poenostavljena arhitektura Android REST odjemalca aplikacije MountainTrips.

RestClient je razred, s katerim v mobilnem delu aplikacije MountainTrips preko interneta dostopamo do strežniškega dela aplikacije Mountaintips oz. izvajamo klice metod REST spletnih storitev.

Razred implementira metodo $getResponse(int, String, Map < String, String >$), ki na podlagi podanih parametrov izvede klic na strežnik in vrne rezultat.

Ker so vse REST metode strežniškega dela aplikacije MountainTrips tipa POST, zna tudi metoda razreda *RestClient* $getResponse(int, String, Map < String, String >)$ izvajati samo poizvedbe tipa POST. V kolikor bi strežniški del aplikacije MountainTrips ponujal tudi druge tipe REST metod, kot so GET, PUT in DELETE, bi morali metodo $getResponse(int, String, Map < String, String >)$ ustrezno razširiti.

Metoda $getResponse(int, String, Map < String, String >)$ pošilja in sprejema podatke iz interneta oz. strežniškega dela aplikacije MountainTrips s pomočjo razreda $HttpURLConnection$. Pošiljanje in sprejemanje podatkov preko interneta poteka z uporabo razreda $HttpURLConnection$ v petih korakih (izvorna koda 5.6):

1. Pridobimo novo instanco $HttpURLConnection$ s klicem $URL.openConnection()$.

2. Nastavimo podatke nove poizvedbe. Poleg podatka URI poizvedbe lahko nastavimo še različne metapodatke.

3. V kolikor želimo pošiljati dodatne parametre, nastavimo instanci $HttpURLConnection$ $setDoOutput(true)$. Podatke zapišemo na tok podatkov (ang. stream) pridobljen z metodo $getOutputStream()$.

4. Rezultat poizvedbe preberemo iz podatkovnega toka, ki ga vrne metoda $getInputStream()$.

5. Z metodo *disconect*() prekinemo povezavo in sprostimo uporabljene vire.

```
URL url = new URL("http://mountaintripsos.rhcloud.com/rest/
    saveData");
HttpURLConnection urlConnection = (HttpURLConnection) url.
    openConnection();
try {
    urlConnection.setDoOutput(true);
    urlConnection.setChunkedStreamingMode(0);

    OutputStream out = new BufferedOutputStream(urlConnection.
        getOutputStream());
    writeStream(out);

    InputStream in = new BufferedInputStream(urlConnection.
        getInputStream());
    readStream(in);
    finally {
        urlConnection.disconnect();
    }
}
```

Izvorna koda 5.6: Primer pošiljanja podatkov na strežnik s pomočjo razreda *HttpURLConnection*.

Android operacijski sistem ne dovoli izvajanja dalj časa trajajočih operacij, kot so pridobivanje GPS lokacije ali klicanje REST metod spletnih storitev, v niti uporabniškega vmesnika (ang. UI thread). Takšne operacije v aplikaciji MountainTrips izvajamo preko storitev (ang. services).

LocationsDataSource

V razredu *LocationsDataSource* so implementirane metode za poizvedovanje in shranjevanje podatkov o lokaciji uporabnika v interno SQLite podatkovno bazo aplikacije MountainTrips (izvorna koda 5.7). Podatkovni model interne podatkovne baze mobilnega dela aplikacije MountainTrips je natančneje predstavljen v poglavju 5.3.10.

```
public Location getLocation(long id) {
    Cursor cursor = database.query(DatabaseHelper.TABLE_LOCATIONS
        , allColums,
    DatabaseHelper.COLUMN_ID + "=" + id, null, null, null, null);
    if(cursor.moveToFirst() != false) {
        Location location = cursorToLocation(cursor);
        cursor.close();
        return location;
    } else {
        return null;
    }
}
```

Izvorna koda 5.7: Primer metode, ki vrne zapis lokacije z danim ID.

RestClientMethods

Razred *RestClientMethods* v statičnih spremenljivkah hrani imena REST metod strežniškega dela aplikacije MountainTrips. Z uporabo statičnih spremenljivk razreda *RestClientMethods* nam v primeru, da se spremeni ime REST metode strežniškega dela aplikacije MountainTrips, ni potrebno spreminjati ime metode po celotnem mobilnem delu aplikacije MountainTrips, ampak samo v razredu *RestClientMethods*.

5.3.5 Paket com.mountain.trips.mobile.helpers

V paketu *com.mountain.trips.mobile.helpers* so razredi, ki implementirajo različne funkcionalnosti. To so posebna orodja, ki v aplikaciji MountainTrips olajšajo postopke kot so:

- Preverjanje dostopnosti do interneta.
- Šifriranje niza znakov z SHA-256 zgoščevalno funkcijo.
- Delo z JSON objekti.
- Nadzor in shranjevanje uporabniških podatkov in podatkov za prijavo uporabnika.

- Shranjevanje nastavitev.
- Validacija, pretvarjanje, zaokroževanje itd.
- Odpiranje, kreiranje in spreminjanje podatkovne baze.

5.3.6 Paket com.mountain.trips.mobile.service

Ker Android operacijski sistem ne dovoli izvajanja dalj časa trajajočih operacij v niti uporabniškega vmesnika, smo v paketu *com.mountain.trips.mobile.service* implementirali razreda *RestService* in *GpsService*, ki razširjata razred *android.app.IntentService*. Razreda predstavljata storitve, ki izvajajo dalj časa trajajoče operacije. Razred *RestService* izvaja dostope do interneta, razred *GpsService* pa skrbi za pridobivanje GPS podatkov lokacije uporabnika.

Storitve implementirane s pomočjo razreda *android.app.IntentService* se izvajajo v delovni niti aplikacije (ang. worker thread), lahko trajajo neomejeno dolgo in se samodejno končajo, ko opravijo svoje naloge. Na enkrat se lahko v delovni niti aplikacije izvaja samo ena storitev tipa *IntentService*.

Vsak razred, ki predstavlja storitev mora biti v *AndroidManifest.xml* ustrezno deklariran z značko < *service* > (izvorna koda 5.8). Izvorna koda 5.9 predstavlja primer klica storitve iz glavne niti.

```
<service android:name="com.mountain.trips.mobile.service.
    RestService" />
<service android:name="com.mountain.trips.mobile.service.
    GpsService" />
```

Izvorna koda 5.8: Deklaracija storitve v datoteki AndroidManifest.xml.

```
final Intent intent = new Intent(this, RestService.class);
intent.putExtra("receiver",this.restWorkerFragment.getReceiver()
    );
intent.putExtra("method",RestClientMethods.GET_MOUNTAIN_NAME);
...
startService(intent);
```

Izvorna koda 5.9: Klic storitve iz glavne niti.

5.3.7 Paket com.mountain.trips.mobile.receiver

Povratno povezavo med storitvami dalj časa trajajočih operacij in aktivnostmi oz. delovnimi fragmenti predstavljata razreda *RestServiceResultReceiver* in *GpsServiceResultReceiver* implementirana v paketu *com.mountain.trips.mobile.receiver*.

Razreda razširjata razred *android.os.ResultReceiver* in imata implementiran vmesnik *Receiver* z definicijo metode $onReceiveResult(int, Bundle)$. Ta metoda se izvede, ko se storitev konča. Metodo implementiramo v aktivnostih ali delovnih fragmentih.

5.3.8 Paket com.mountain.trips.mobile.workerfragments

Povezava med storitvami dalj časa trajajočih operacij in aktivnostmi s pomočjo vmesnih sprejemnikov (ang. receiver) implementiranih v paketu *com.mountain.trips.mobile.receiver* se izgubi, če med izvajanjem storitve uporabnik zaslon zavrti oz. če aplikacija spremeni izris uporabniškega vmesnika med pokrajino in portretom ali obratno.

Pri vrtenju zaslona Android operacijski sistem uniči staro aktivnost, ki prikazuje aplikacijo in ustvari novo instanco aktivnosti [41]. Če se med izvajanjem dalj časa trajajoče storitve zaslon zavrti, se aktivnost, ki je podala zahtevo za izvedbo storitve, uniči. V tem primeru vmesni sprejemnik nima informacije o uničenju aktivnosti in pošlje rezultate storitve že uničeni instanci

aktivnosti. Nova instanca aktivnost, ki je ustvarjena pri vrtenju zaslona, pa nikoli ne prejme rezultatov poizvedbe.

Problem povezave med aktivnostmi in storitvami v primeru vrtenja zaslona smo rešili z uporabo fragmentov (ang. Fragment) brez uporabniškega vmesnika, kjer ostanejo rezultati storitev shranjeni tudi v primeru, ko se zaslon zavrti. Takšnim tipom fragmentov pravimo tudi delovni fragmenti (ang. worker fragment) [56].

V mobilnem delu aplikacije MountainTrips predstavljajo dalj časa trajajoče operacije dostopi do interneta oz. REST metod spletnih storitev strežniškega dela aplikacije MountainTrips in dostop do GPS lokacije uporabnika. Dalj časa trajajoče operacije izvajajo storitve, ki so implementirane v razredih *RestService* in *GPSService*. Razreda, ki predstavljata storitve, sta vključena v paket *com.mountain.trips.mobile.service*. Storitve vračajo rezultate preko vmesnih sprejemnikov in delovnih fragmentov nazaj k aktivnostim.

V aplikaciji MountainTrips sta implementirana dva delovna fragmenta: $RestWorkerFragment$ in $GPSWorkerFragment$. Vključena sta v paketu $com.mountain.trips.mobile.worker fragments$.

5.3.9 Uporabniški vmesnik

V aplikacijah za operacijski sistem Android lahko definiramo uporabniški vmesnik neposredno v izvorni kodi aplikacije in s pomočjo XML datotek. Možna je tudi kombinacija obeh pristopov. To pomeni, da osnovno strukturo uporabniškega vmesnika definiramo v datotekah XML, nato pa jo v izvorni kodi aplikacije natančneje prilagodimo.

Kombinacijo obeh pristopov smo uporabili tudi v aplikaciji MountainTrips. V datotekah XML, ki se nahajajo v mapi $/MountainTrips/res/layout$, smo definirali postavitev in elemente uporabniškega vmesnika, ki jih v izvorni kodi aplikacije natančneje nastavljamo. Elementom nastavljamo njihovo vidnost in aktivnost (omogočen/onemogočen).

5.3.10 Podatkovna baza mobilnega dela aplikacije MountainTrips

V primeru, da aplikacija MountainTrips nima dostopa do interneta, se GPS podatki lokacij shranijo v lokalno SQLite podatkovno bazo.

Lokalna podatkovna baza mobilnega dela aplikacije MountainTrips je preprosta. Sestavlja jo tabela *locations*, z naslednjimi atributi:

accurancy Natančnost podatkov o lokaciji.

altitude Nadmorska višina lokacije, kjer se je uporabnik nahajal.

latitude Zemljepisna širina lokacije, kjer se je uporabnik nahajal.

longitude Zemljepisna dolžina lokacije, kjer se je uporabnik nahajal.

comment Komentar.

recordDate Datum, ko se je uporabnik nahajal na lokaciji, ki jo zapis o lokaciji predstavlja.

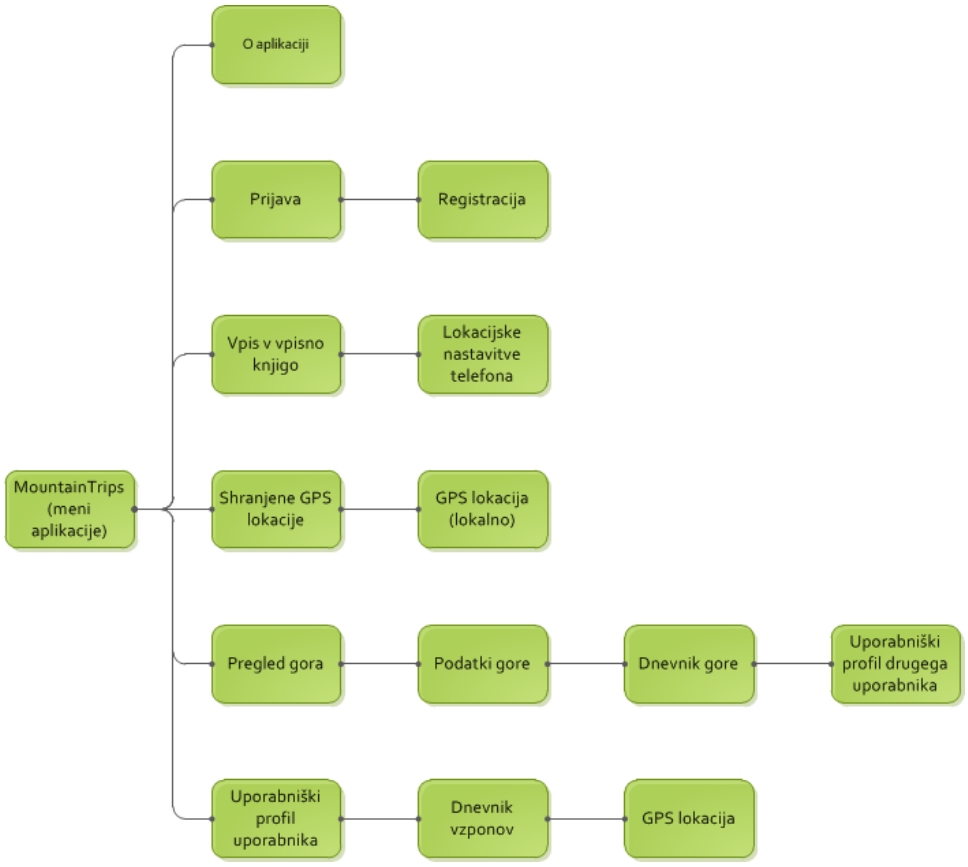
Poglavje 6

Predstavitev aplikacije MountainTrips

Aplikacija MountainTrips s svojimi funkcionalnostmi povzema lastnosti klasične vpisne knjige vrhov gora in osebnega dnevnika vzponov. Hkrati omogoča uporabnikom deljenje kontaktnih podatkov med seboj. S tem želimo ljubitelje gora med seboj tesneje povezati v aktivno skupino ljudi, ki med sabo deli uporabne informacije o gorah in gorskih poteh.

Mobilna aplikacija MountainTrips je razvita za operacijski sistem Android različice vsaj 2.2.

V nadaljevanju so predstavljene funkcionalnosti aplikacije. Določene funkcionalnosti so omogočene vedno, nekatere pa delujejo le, če ima aplikacija dostop do interneta in če je uporabnik v aplikacijo prijavljen. Slika 6.1 prikazuje medsebojno povezavo pogledov oz. funkcionalnosti aplikacije, slika 6.2 pa UML diagram primerov uporabe mobilnega dela aplikacije MountainTrips.

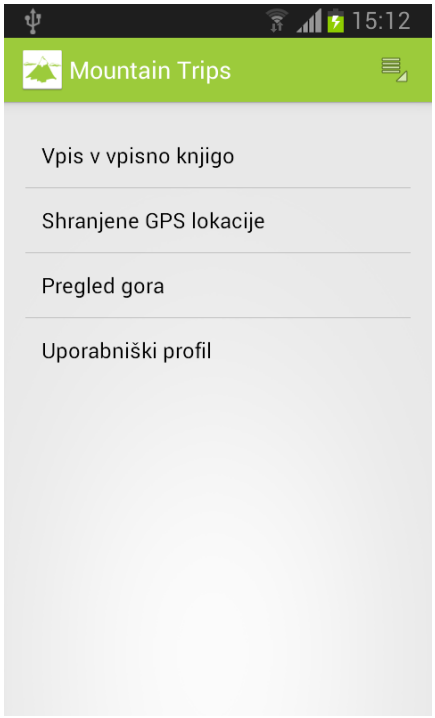


Slika 6.1: Pogledi aplikacije MountainTrips.

Slika 6.2: UML diagram primerov uporabe mobilnega dela aplikacije MountainTrips.

6.1 Meni aplikacije MountainTrips

Ob zagonu aplikacije MountainTrips se odpre meni s štirimi gumbi, ki nas vodijo v različne funkcionalnosti aplikacije (slika 6.3). V primeru, da aplikacija nima dostopa do interneta ali uporabnik aplikacije ni prijavljen, so določene funkcionalnosti aplikacije omejene, zato so določeni gumbi menija aplikacije onemogočeni.

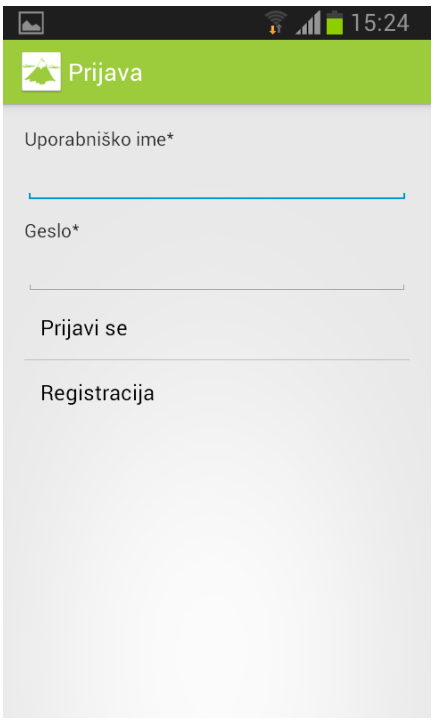


Slika 6.3: Meni aplikacije MountainTrips.

V zgornjem desnem kotu menija aplikacije je gumb, ki odpre spustni meni. Spustni meni vsebuje povezavo na prijavo (slika 6.4) oz. odjavo in povezavo na pogled z osnovnimi podatki o aplikaciji (slika 6.16).

6.2 Registracija in prijava

Določene funkcionalnosti aplikacije zahtevajo prijavo uporabnika. Uporabnik se prijavi v aplikacijo z uporabniškim imenom in geslom, ki ju določi ob registraciji. V primeru, da uporabnik vnese napačne podatke, se izpiše obvestilo z opisom napake. Pogled za prijavo uporabnika je prikazan na sliki 6.4.

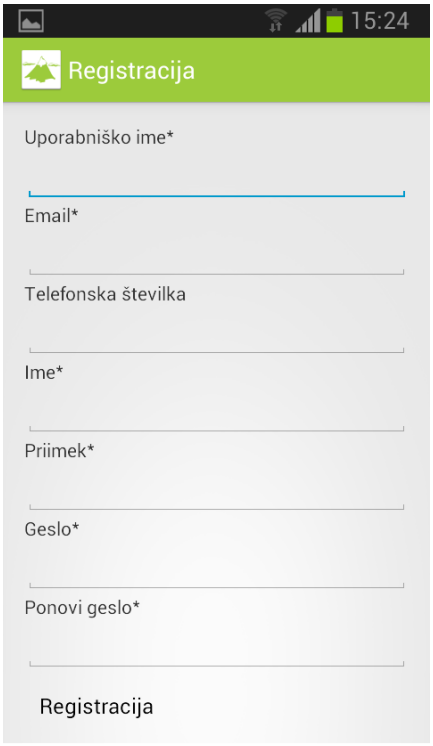


Slika 6.4: Prijava uporabnika.

Gumb *Registracija* na pogledu za prijavo uporabnika odpre pogled za registracijo novega uporabnika (slika 6.5). Ob registraciji vnese uporabnik naslednje podatke:

- uporabniško ime,
- naslov elektronske pošte,
- telefonsko številko,

- ime,
- priimek,
- geslo

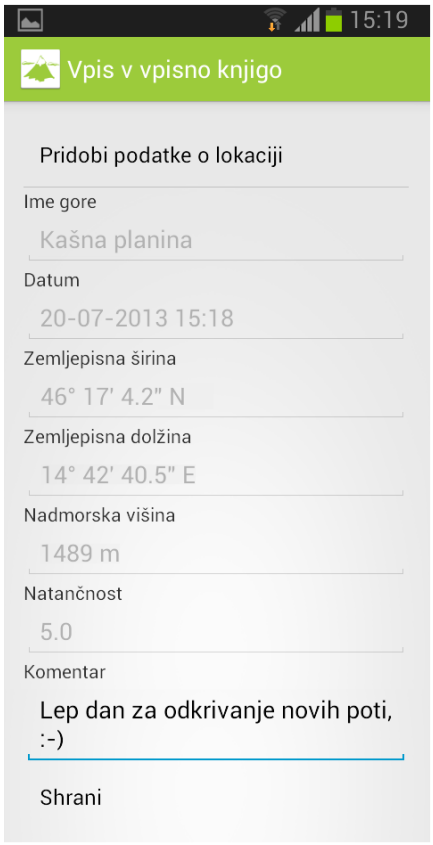


Slika 6.5: Registracija novega uporabnika.

Podatki, ki so za registracijo obvezni, so označeni z $*$. Uporabnik potrdi registracijo s klikom na gumb *Registracija*. Če so vneseni podatki pravilni, se izpiše obvestilo o uspešno opravljeni registraciji novega uporabnika, sicer pa obvestilo z opisom napake.

6.3 Vpis v vpisno knjigo

Vpis v vpisno knjigo omogoča pridobivanje in shranjevanje podatkov o lokaciji, na kateri se nahaja uporabnik (slika 6.6).



Slika 6.6: Vpis v vpisno knjigo.

S klikom na gumb *Pridobi podatke o lokaciji* začne aplikacija s pridobivanjem GPS podatkov o lokaciji, na kateri se uporabnik nahaja. V primeru da, aplikacija uspešno pridobi GPS podatke o lokaciji uporabnika, se samodejno izpolnijo naslednja polja:

- datum,
- zemljepisna širina,

- zemljepisna dolžina,
- nadmorska višina,
- natančnost

Če ima aplikacija dostop do interneta, poskuša na podlagi pridobljenih GPS podatkov o lokaciji ugotoviti ime gore, kjer se uporabnik nahaja. V kolikor pridobljeni GPS podatki ustrezajo lokaciji vrha gore, ki je shranjena v bazi na strežniku, se samodejno izpolni polje *ime gore*.

Za pridobivanje GPS podatkov potrebuje aplikacija MountainTrips vključeno sprejemanje GPS signala. Če sprejemanje GPS signala ni vključeno, se uporabniku izpiše obvestilo, ki ga napoti v lokacijske nastavitve telefona, kjer lahko uporabnik vključi sprejemanje podatkov o lokaciji iz GPS satelitov.

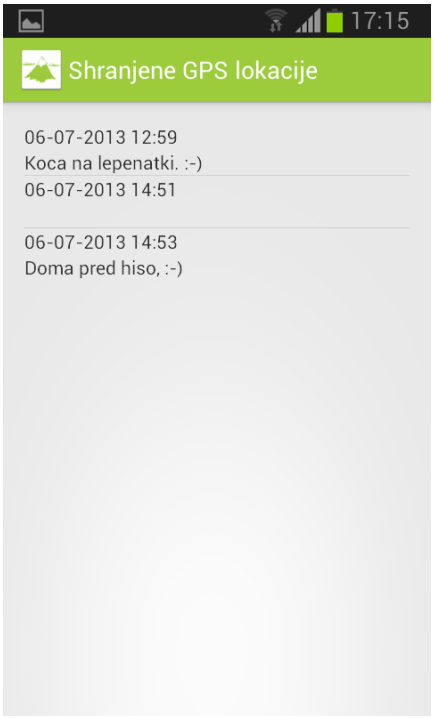
Pridobivanje GPS podatkov je omejeno na tri minute. Če aplikacija ne prejme GPS podatkov o lokaciji v času treh minut, se pridobivanje GPS podatkov prekine. V tem primeru se izpiše obvestilo z vprašanjem, če želi uporabnik ponoviti ali prekiniti pridobivanje GPS podatkov o lokaciji uporabnika.

Pogled *Vpis v vpisno knjigo* vključuje še polje *Komentar* in gumb *Shrani*. V polje *Komentar* lahko uporabnik napiše poljuben komentar, s klikom na gumb *Shrani* pa shrani podatke v lokalno podatkovno bazo mobilne aplikacije MountainTrips oz. na strežnik.

Če ima aplikacija MountainTrips dostop do interneta in če je uporabnik v aplikacijo prijavljen, se s klikom na gumb *Shrani* podatki o lokaciji uporabnika samodejno shranijo na strežnik. Sicer se podatki shranijo v lokalno podatkovno bazo mobilne aplikacije MountainTrips, uporabnik pa jih lahko kasneje, ko ima dostop do interneta, shrani na strežnik.

6.4 Shranjene GPS lokacije

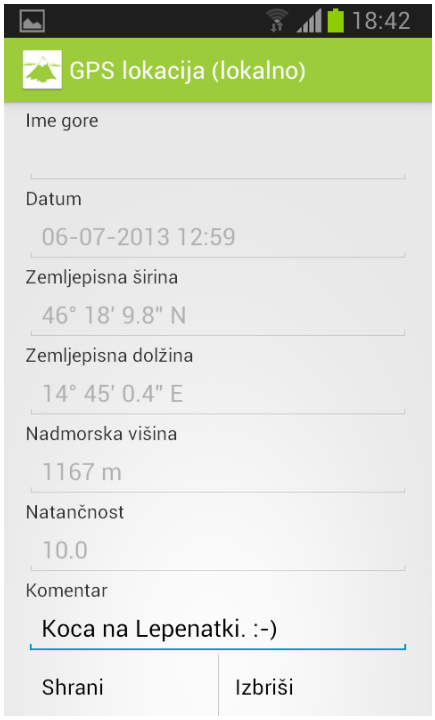
Pogled *Shranjene GPS lokacije* predstavlja seznam GPS lokacij, shranjenih v lokalno podatkovno bazo aplikacije MountainTrips, na katerih se je nahajal uporabnik (slika 6.7).



Slika 6.7: Shranjene GPS lokacije.

Elementi seznama predstavljajo datum zapisa pridobljenih GPS podatkov o lokaciji uporabnika v lokalno podatkovno bazo in komentar. S klikom na element seznama odpremo pogled *GPS lokacija* (*lokalno*) (slika 6.8), ki prikazuje lokalno shranjene GPS podatke o lokaciji, na kateri se je nahajal uporabnik in komentar. V primeru, da je aplikacija povezana z internetom, poskuša na podlagi shranjenih GPS podatkov o lokaciji pridobiti ime gore, katere lokacijski podatki vrha se ujemajo s shranjenimi GPS podatki o lokaciji. V primeru ujemanja aplikacija MountainTrips samodejno izpolni

polje *ime gore*.



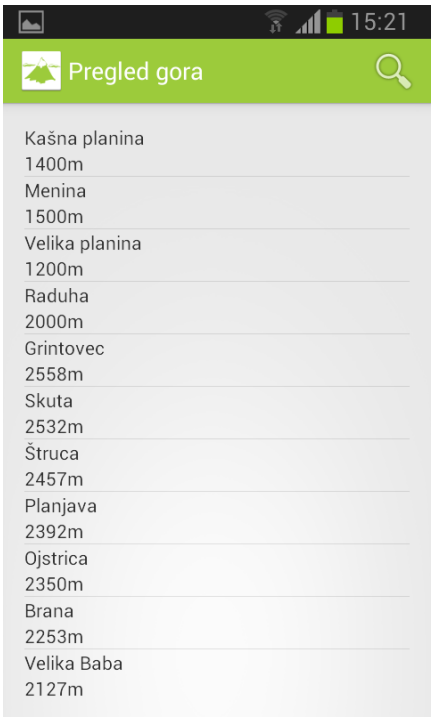
Slika 6.8: GPS lokacija (lokalno).

S klikom na gumb *Shrani* shranimo GPS podatke o lokaciji, pridobljeno ime lokacije in komentar. Če ima aplikacija dostop do interneta, se podatki shranijo na strežnik, sicer pa nazaj v lokalno podatkovno bazo. Gumb *Izbriši* izbriše GPS podatke o lokaciji iz lokalne podatkovne baze aplikacije MountainTrips.

6.5 Pregled gora

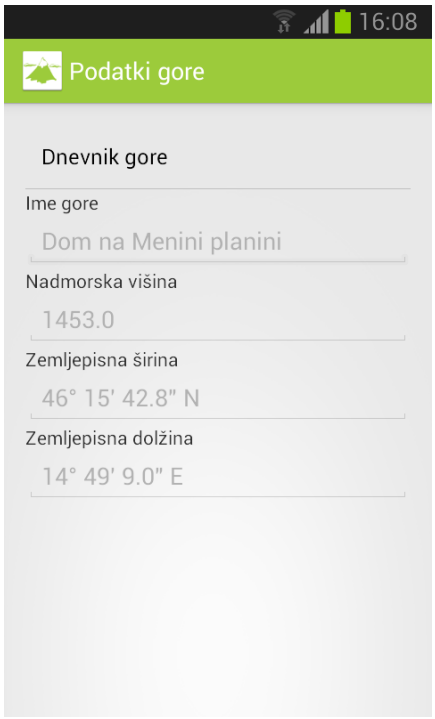
Pregled gora prikazuje seznam gora in omogoča njihovo iskanje (slika 6.9). Vsak element seznama prikazuje ime gore in njeno nadmorsko višino.

Za iskanje in pregledovanje gora ter njihovih podatkov mora biti uporabnik prijavljen in imeti dostop do interneta.



Slika 6.9: Pregled in iskanje gora.

S klikom na posamezen element seznama gora odpremo pogled, ki prikazuje podrobnejše podatke o izbrani gori (slika 6.10) in povezavo na dnevnik gore. Dnevnik gore predstavlja vpise uporabnikov v vpisno knjigo vrha gore (slika 6.11).

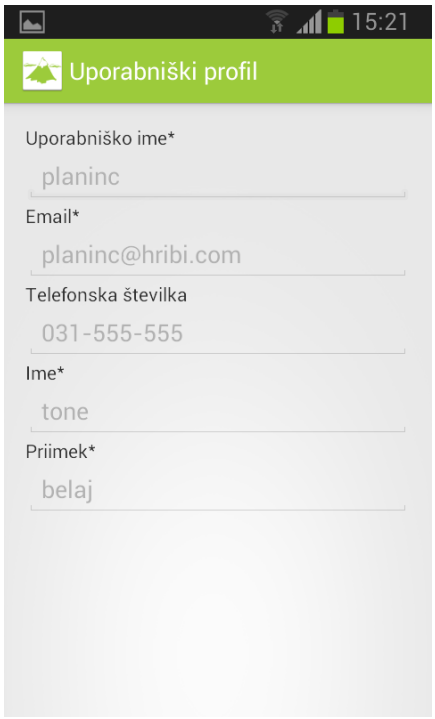


Slika 6.10: Podatki gore.

Slika 6.11: Dnevnik gore.

Dnevnik gore je seznam vpisov planincev na vrhu gore preko aplikacije MountainTrips. To so vpisi, pri katerih je aplikacija MountainTrips na podlagi GPS podatkov o lokaciji sama ugotovila ime gore. Vsak element seznama dnevnika gore je sestavljen iz uporabniškega imena, komentarja in točnega časa, ko se je planinec nahajal na vrhu gore.

Klik na element seznama vpisov dnevnika gore odpre podrobnejše informacije o planincu oz. uporabniku aplikacije, kateremu pripada izbran vpis v dnevnik vrha gore (Slika 6.12).

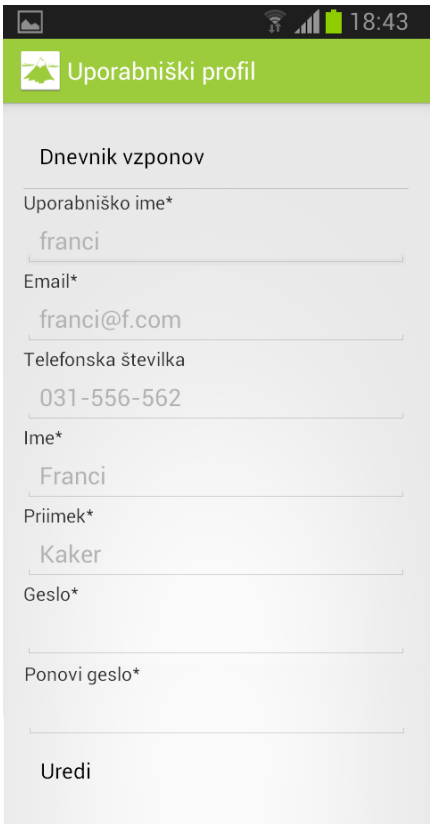


Slika 6.12: Pregled podatkov planinca.

6.6 Uporabniški profil

Pogled *Uporabniški profil* vsebuje povezavo na dnevnik vzponov uporabnika ter omogoča pregled in urejanje podatkov uporabniškega profila (slika 6.13). Uporabnik lahko nastavi:

- naslov elektronske pošte,
- telefonsko številko,
- ime,
- priimek,
- geslo



Slika 6.13: Pregled in urejanje podatkov uporabnika.

S klikom na gumb *Dnevnik vzponov* odpremo dnevnik vzponov uporabnika (slika 6.14). To je seznam vseh shranjenih GPS lokacij, ki jih je uporabnik shranil na strežnik, ne glede na to, ali je lokacija shranjena v bazi vrhov gora aplikacije MountainTrips ali ne. Vsak element seznama dnevnika vzponov uporabnika je sestavljen iz točnega časa, ko so bili GPS podatki o lokaciji pridobljeni, komentarja uporabnika in imena gore, če je aplikacija MountainTrips prepoznala ime gore glede na GPS podatke o lokaciji.

Slika 6.14: Dnevnik vzponov uporabnika.

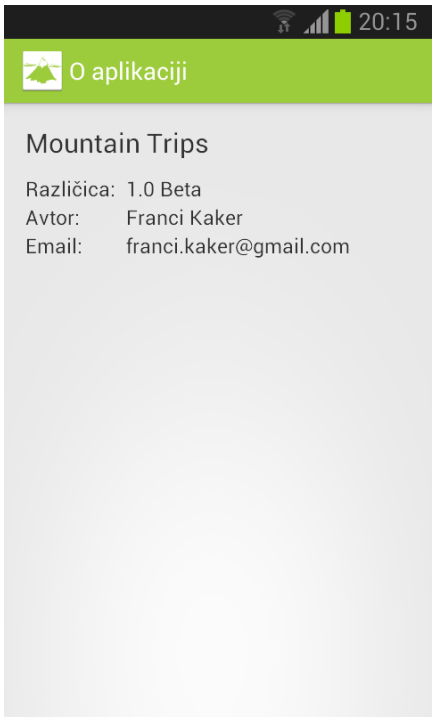
S klikom na element seznama dnevnika vzponov uporabnika odpremo podrobnejšo predstavitev GPS podatkov o lokaciji posameznega uporabniškega vpisa na strežnik (slika 6.15).

Slika 6.15: GPS lokacija.

Za pregled uporabniških podatkov mora biti uporabnik prijavljen, ne potrebuje pa dostopa do interneta. V primeru, da želi uporabnik urejati uporabniške podatke in pregledovati lasten dnevnik vzponov, pa mora biti prijavljen in imeti dostop do interneta.

6.7 Podatki o aplikaciji MountainTrips

Pogled *O aplikaciji* prikazuje različico aplikacije in podatke o avtorju aplikacije (slika 6.16).



Slika 6.16: Podatki o aplikaciji MountainTrips.

Poglavje 7

Možne izboljšave in razširitve

Med izdelavo aplikacije MountainTrips smo velik poudarek namenili načrtovanju arhitekture aplikacije. Zaradi želje po izdelavi zanesljive aplikacije smo raje implementirali kakšno funkcionalnost manj, implementirane funkcionalnosti pa izdelali kvalitetno. Zato bi pri nadaljnjem razvoju usmerili več energije in pozornosti v razširitev funkcionalnosti aplikacije in manj v izboljšave na področju tehnične implementacije aplikacije MountainTrips.

Glede na to, da smo v okviru diplomske naloge v aplikacijo MountainTrips vključili zgolj osnovne funkcionalnosti klasične vpisne knjige in osebnega dnevnika vzponov planinca, je možnosti za razširitev funkcionalnosti aplikacije MountainTrips precej.

Ob nadaljnjem razvoju aplikacije lahko osnovnim podatkom planin dodamo podrobnejše opise planin, planinskih poti in sliko žiga vrha gore. K opisom planin in planinskih poti bi lahko dodali še vremensko napoved in podatke o trenutnih vremenskih razmerah na planinah.

Implementirali bi lahko komentiranje gora in planinskih poti ter dodajanje slik planin, planinskih poti in razglednih točk.

Uporabniškemu profilu aplikacije MountainTrips se lahko dodajo še podatki, kot so naslov bivanja, članstvo v katerem od planinskih društev, ime planinskega društva itd. Hkrati bi dodali možnost nastavljanja vidnosti osebnih podatkov javnosti oz. drugim uporabnikom. Uporabniški profil se lahko

razširi še s pogledom, ki prikazuje žige vrhov gora, ki jih je osvojil uporabnik aplikacije MountainTrips.

Iz vpisnih podatkov vrhov gora lahko izluščimo različne statistične podatke, ki bi jih lahko prikazali v opisih gora. Prikazovali bi lahko npr. seznam uporabnikov, ki so najpogosteje osvojili vrh določene planine, najbolj priljubljene poti na planino itd.

Tudi uporabnikom bi lahko v uporabniški profil dodali seznam najpogosteje osvojenih vrhov.

Kljub temu, da so vse omenjene razširitve funkcionalnosti z malo truda izvedljive, bi bilo pametno aplikacijo MountainTrips čim hitreje ponuditi uporabnikom ter nato implementirati razširitve in funkcionalnosti glede na njihove želje in potrebe.

Prvi stik uporabnika z aplikacijo je zelo pomemben, saj uporabnika pogosto odvrne ali pritegne k uporabi aplikacije. Zato je toliko bolj pomembno, da ponuja aplikacija prijetno uporabniško izkušnjo.

Aplikacija MountainTrips, izdelana v okviru diplomske naloge še ni izdelana do stopnje, da bi jo lahko ponudili končnim uporabnikom. Na področju tehnične implementacije bi bilo treba premisliti o smiselnosti implementacije povezave med strežnikom in mobilnim odjemalcem s pomočjo kriptografskih protokolov, kot sta SSL ali TLS in jo po potrebi implementirati. Uporabnik bi si lahko nato sam izbral ali želi pošiljati podatke po šifrirani povezavi ali ne.

Aplikaciji MountainTrips bi lahko še stilsko izboljšali uporabniški vmesnik, podporo za različne velikosti zaslonov in podporo vrtenju zaslonov.

Poglavje 8

Sklepne ugotovitve

V okviru diplomske naloge smo implementirali aplikacijo MountainTrips. Aplikacija MountainTrips deluje na mobilnih telefonih, ki poganjajo operacijski sistem Android ter vključuje osnovne funkcionalnosti klasične vpisne knjige in osebnega dnevnika vzponov.

Za vpis v elektronsko vpisno knjigo vrha gore preko aplikacije MountainTrips potrebujemo delujoč mobilni telefon z operacijskim sistemom Android, GPS sprejemnikom in dostop do interneta. Na sprejem GPS signala in dostop do interneta lahko negativno vplivajo različni vremenski vplivi, ki nam v najslabšem primeru onemogočijo vpis v elektronsko vpisno knjigo. Zato ni namen aplikacije MountainTrips popolnoma nadomestiti klasično vpisno knjigo vrhov gora.

Aplikacija MountainTrips razvita v okviru diplomske naloge vključuje zgolj nekatere funkcionalnosti večjega sistema, ki bi uporabnikom omogočal preprostejše medsebojno povezovanje in deljenje informacij o gorah in gorskih poteh. Nadaljnji razvoj aplikacije MountainTrips je opisan v poglavju 7.

Zavedamo se, da lahko napake povzročene v začetnih fazah programske opreme, vplivajo na preprostost nadaljnjega razvoja in zanesljivost aplikacije. Ker ima aplikacija MountainTrips velik razširitveni potencial, smo pri razvoju namenili veliko časa izboru tehnologij in implementaciji aplikacije.

Aplikacija MountainTrips je sestavljena iz strežniškega in mobilnega dela

aplikacije. Tehnologije za razvoj strežniškega dela aplikacije smo že poznali, zato nam implementacija strežniškega dela aplikacije ni predstavljala večjih težav. Ker smo imeli z razvojem aplikacij za Android operacijski sistem pred izdelavo aplikacije MountainTrips relativno malo izkušenj, smo imeli veliko več težav z načrtovanjem in razvojem mobilnega dela aplikacije MountainTrips. Na težave smo naleteli pri implementaciji mobilnega odjemalca REST spletnih storitev, implementaciji podpore vrtenju zaslonov in z nepopolno kompatibilnostjo novejših programskih vmesnikov s starejšimi različicami Android operacijskega sistema.

Pomembne izkušnje smo pridobili tudi z uporabo OpenShift PaaS oblačne storitve, kjer teče strežniški del aplikacije MountainTrips. Uporaba oblačnih storitev ima tako prednosti kot slabosti. Ena od prednosti je, da povečujejo zanesljivost delovanja aplikacije v primeru nenadnega povečanja prometa, ena od pomembnih slabosti, glede na naše izkušnje, pa je, da uporabnik oblačne storitve nima popolnega nadzora nad izvajanjem aplikacije. Ponudnik oblačnih storitev jo lahko med različnimi opravili, kot so različna vzdrževanja in posodobitve, prekine oz. ustavi.

Celoten čas razvoja aplikacije MountainTrips je bil zaradi različnih težav med razvojem mobilnega dela aplikacije MountainTrips daljši od načrtovanega. Vse težave sem sprejemal kot izziv in jih poskušal rešiti kar najbolj pravilno in kvalitetno. Menim, da sem na ta način pridobil veliko novega znanja in izkušenj, ki mi bodo koristile pri nadaljnjem delu in prihodnjih projektih.

Literatura

[1] "Eclipse (software)" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/Eclipse_(software)

[2] "JBoss Tools" (2013, Julij) Dostopno na:
http://www.jboss.org/tools

[3] "JBoss Tools" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/JBoss_Tools

[4] "Elektronska vpisna knjiga portala hribi.net" (2013, Junij) Dostopno na:
http://www.hribi.net/eknjiga

[5] "Hill Lists" (2013, Junij) Dostopno na:
http://grahamhaley.co.uk/hills/

[6] "Java (programming language)" (2013, Junij) Dostopno na:
https://en.wikipedia.org/wiki/Java_(programming_language)

[7] "Java virtual machine" (2013, Junij) Dostopno na:
https://en.wikipedia.org/wiki/Java_virtual_machine

[8] Michael Barr "KVM: A Small Java Virtual Machine for J2ME" (2013, Junij) Dostopno na:
https://en.wikipedia.org/wiki/Java_virtual_machine

[9] dr. Matjaž B. Jurič "Prosojnice za predmet Postopki razvoja programske opreme" (2013, Julij) Dostopno na: https://ucilnica.fri.uni-lj.si/mod/resource/view.php?id=19764

[10] "Java Platform, Enterprise Edition" (2013, Julij) Dostopno na: http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition

[11] C. Bauer, G. King, Java Persistence with Hibernate. Manning Publications, 2006.

[12] "Hibernate (Java)" (2013, Julij) Dostopno na: http://en.wikipedia.org/wiki/Hibernate_(Java)

[13] "MySQL" (2013, Julij) Dostopno na: http://sl.wikipedia.org/wiki/MySQL

[14] "MySQL" (2013, Julij) Dostopno na: https://en.wikipedia.org/wiki/MySQL

[15] "MySQL" (2013, Julij) Dostopno na: www.mysql.com

[16] "RESTEasy documentation" (2013, Julij) Dostopno na: http://docs.jboss.org/resteasy/docs/2.3.6.Final/userguide/html_single/index.html

[17] "JSON" (2013, Julij) Dostopno na: http://www.json.org/

[18] "JSON" (2013, Julij) Dostopno na: http://en.wikipedia.org/wiki/JSON

[19] "Google GSON" (2013, Julij) Dostopno na: http://code.google.com/p/google-gson/

[20] "WildFly" (2013, Julij) Dostopno na: http://en.wikipedia.org/wiki/WildFly

[21] "OpenShift" (2013, Maj) Dostopno na:
https://www.openshift.com/

[22] "OpenShift technologies" (2013, Julij) Dostopno na:
https://www.openshift.com/developers/technologies

[23] "OpenShift" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/OpenShift

[24] "The 2nd Tenet of PaaS: The P in PaaS Stands for Platform, Not Product" (2013, Junij) Dostopno na:
http://planet.jboss.org/post/the_2nd_tenet_of_paas_the_p_in_paas_stands_for_platform_not_product

[25] "Google App Engine" (2013, Maj) Dostopno na:
https://developers.google.com/appengine/

[26] "Apache Maven" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/Apache_Maven

[27] M. Kalin, Java Web Services: Up and Running. O'Reilly, 2009.

[28] "Web service" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/Web_service

[29] B. Burke, RESTful Java with JAX-RS. O'Reilly, 2009.

[30] "REST" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/REST

[31] "REST APIs must be hypertext-driven" (2013, Julij) Dostopno na:
http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

[32] "SOAP" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/SOAP

[33] "Mobile phone" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/Mobile_phone

[34] "Android (operating system)" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/Android_(operating_system)

[35] "Android Architecture" (2013, Julij) Dostopno na:
http://www.android-app-market.com/android-architecture.html

[36] "Google: There Are 900 Million Android Devices Activated" (2013, Julij) Dostopno na:
http://www.businessinsider.com/900-million-android-devices-in-2013-2013-5

[37] "Android software development" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/Android_software_development

[38] "ActionBarSherlock" (2013, Julij) Dostopno na:
http://actionbarsherlock.com/

[39] "SQLite" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/SQLite

[40] "SQLite" (2013, Julij) Dostopno na:
http://www.sqlite.org

[41] J. Steele, N. To, The Android Developer's Cookbook Building Applications with the Android SDK. Addison-Wesley, 2010.

[42] "Android Developer" (2013, Julij) Dostopno na:
http://developer.android.com/index.html

[43] "Android Developer Tools" (2013, Julij) Dostopno na:
http://developer.android.com/tools/help/adt.html

[44] "Support Library" (2013, Julij) Dostopno na:
http://developer.android.com/tools/support-library/index.html

[45] "Android Developer" (3. 9. 2013) Dostopno na:
http://developer.android.com/about/dashboards/index.html

[46] A. El-Rabbany, Introduction to GPS The Global Positioning System. Artech House, 2002.

[47] "GPS" (2013, Julij) Dostopno na:
http://sl.wikipedia.org/wiki/GPS

[48] "Global Positioning System" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/Global_Positioning_System

[49] "EAR (file format)" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/EAR_

[50] "WAR (file format)" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/WAR_

[51] "Java API for RESTful Web Services" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/Java_API_for_RESTful_Web_Services

[52] "Activity lifecycle" (2013, Julij) Dostopno na:
http://www.anddev.org/images/android/activity_lifecycle.png

[53] "APK (file_format)" (2013, Julij) Dostopno na:
http://en.wikipedia.org/wiki/APK_(file_format)

[54] "Google I/O 2010 - Android REST client applications" (2013, Maj) Dostopno na:
http://youtu.be/xHXn3Kg2IQE

[55] "Google I/O 2010 - Android REST client applications (PDF)" (2013, Maj) Dostopno na:
http://dl.google.com/googleio/2010/android-developing-RESTful-android-apps.pdf

[56] "Using a WorkerFragment" (2013, Maj) Dostopno na:
http://zapek.com/blog/using-a-workerfragment/